Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

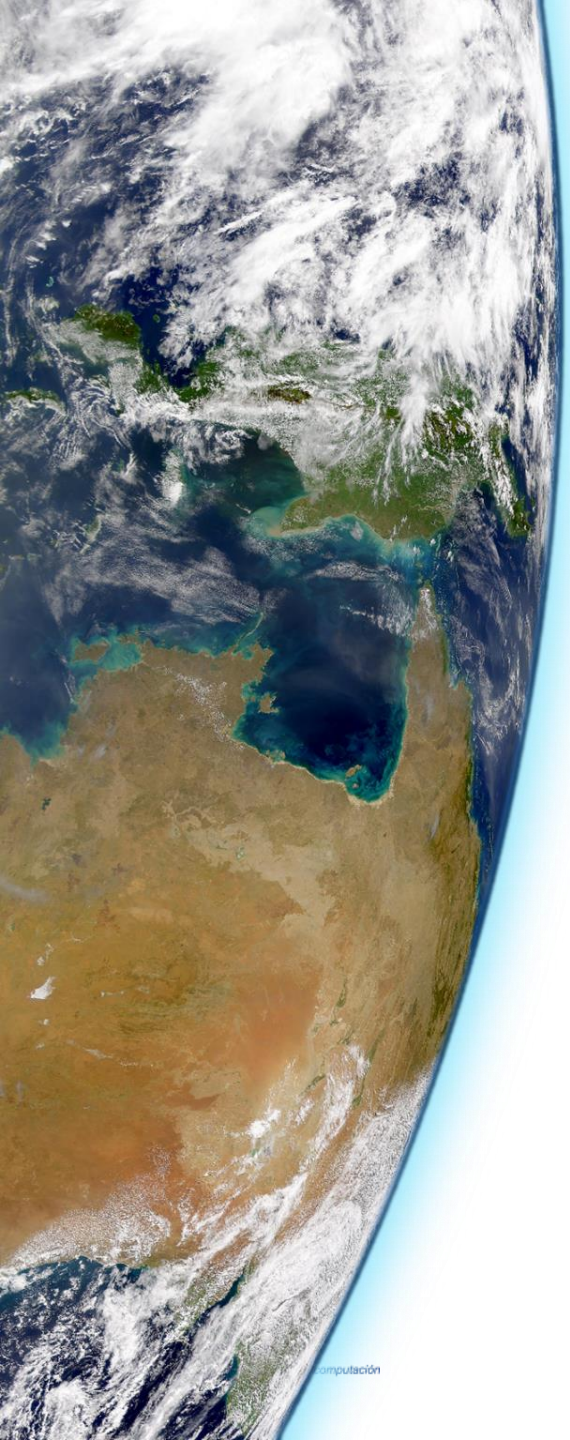
Improving Ocean Model Computational Performance by using Mixed-Precision Approaches

Oriol Tintó, Mario Acosta, Miguel Castrillo, Kim Serradell, Francisco J. Doblas-Reyes

Computational Earth Sciences

PASC19 - High-Resolution Weather and Climate Simulations

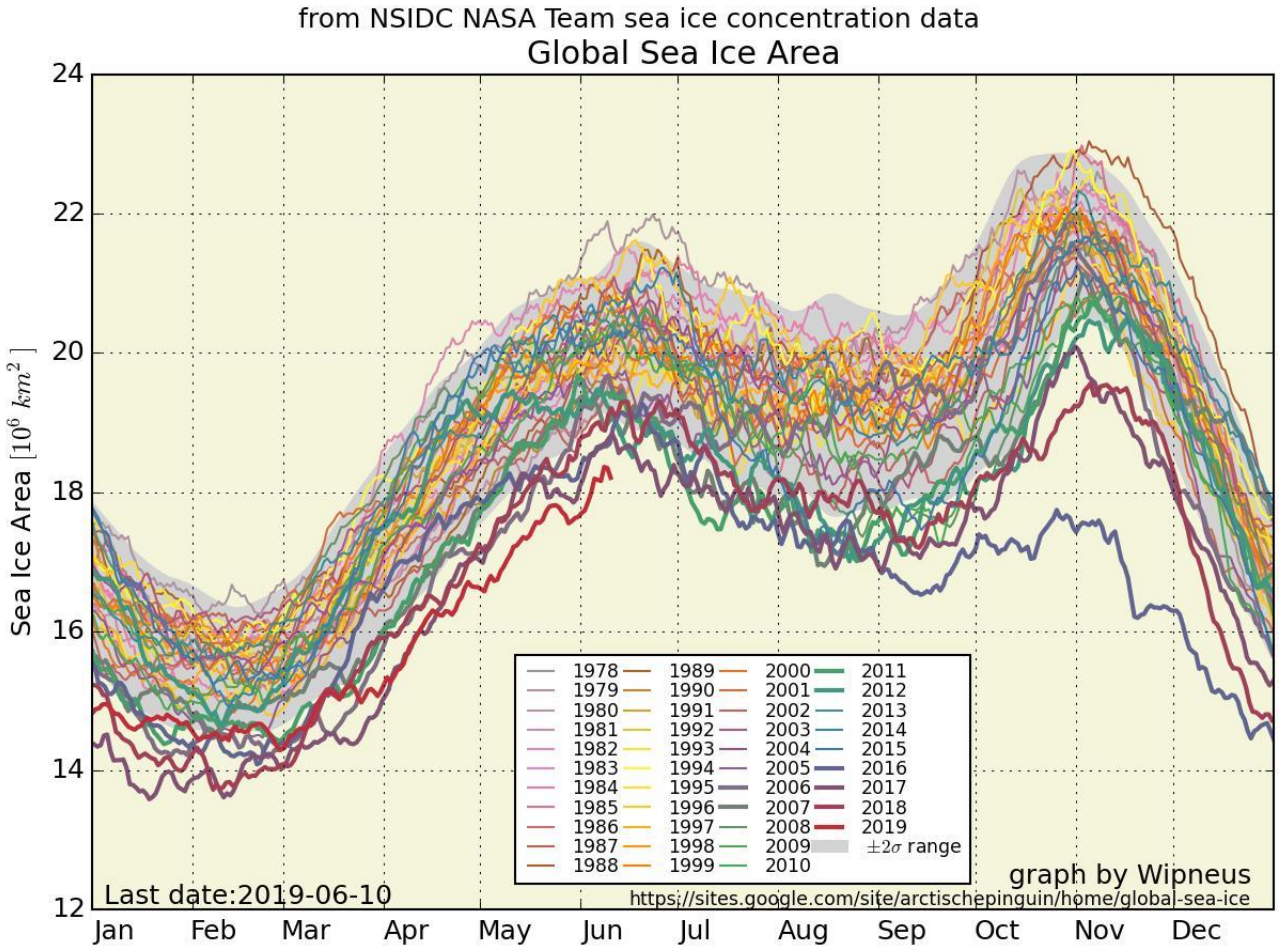
14/06/2019



Outline

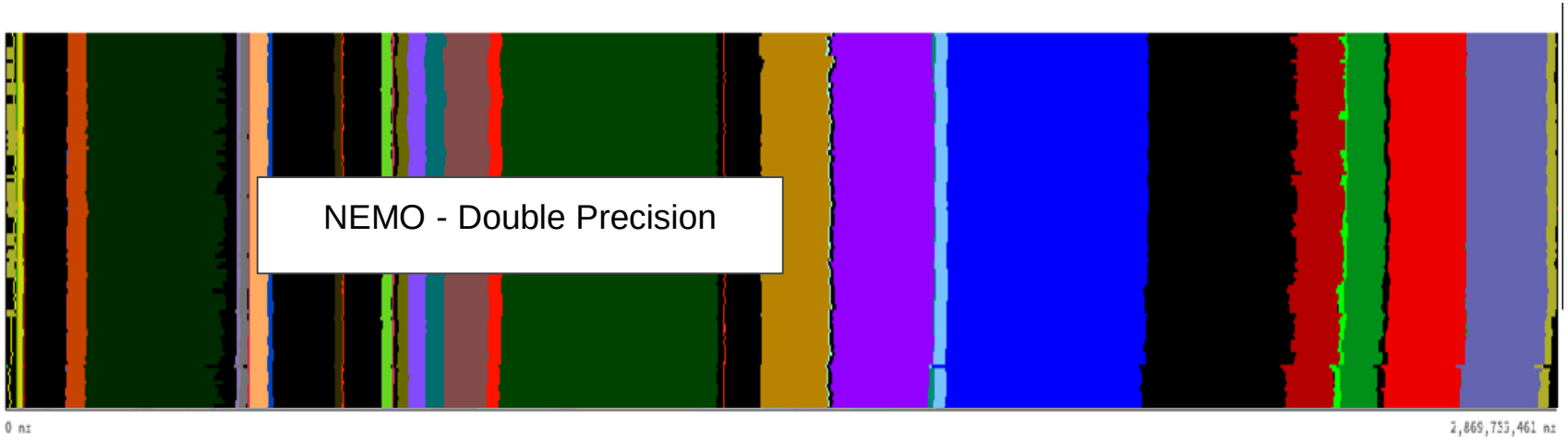
- Why?
- How?

Why models?

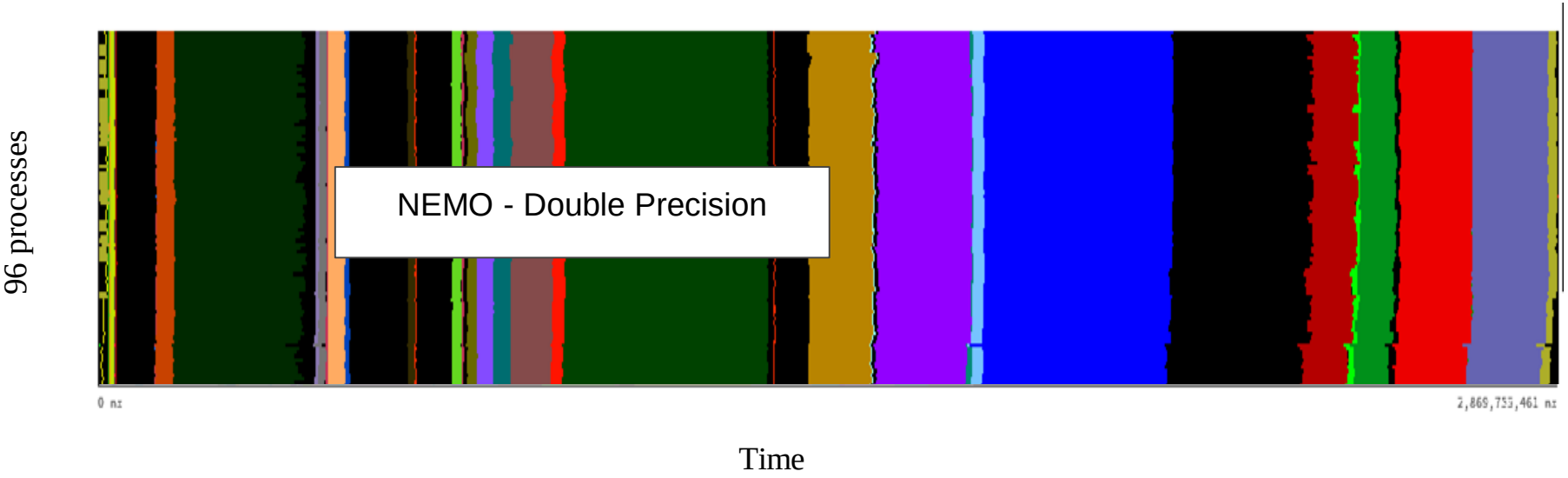

from NSIDC NASA Team sea ice concentration data
Global Sea Ice Area
Sea Ice Area [10^6 km^2]
24
22
20
18
16
14
12
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
1978 1989 2000 2011
1979 1990 2001 2012
1980 1991 2002 2013
1981 1992 2003 2014
1982 1993 2004 2015
1983 1994 2005 2016
1984 1995 2006 2017
1985 1996 2007 2018
1986 1997 2008 2019
1987 1998 2009 $\pm 2\sigma$ range
1988 1999 2010
Last date:2019-06-10
graph by Wipneus
https://sites.google.com/site/arctischepinguin/home/global-sea-ice

Why mixed precision?

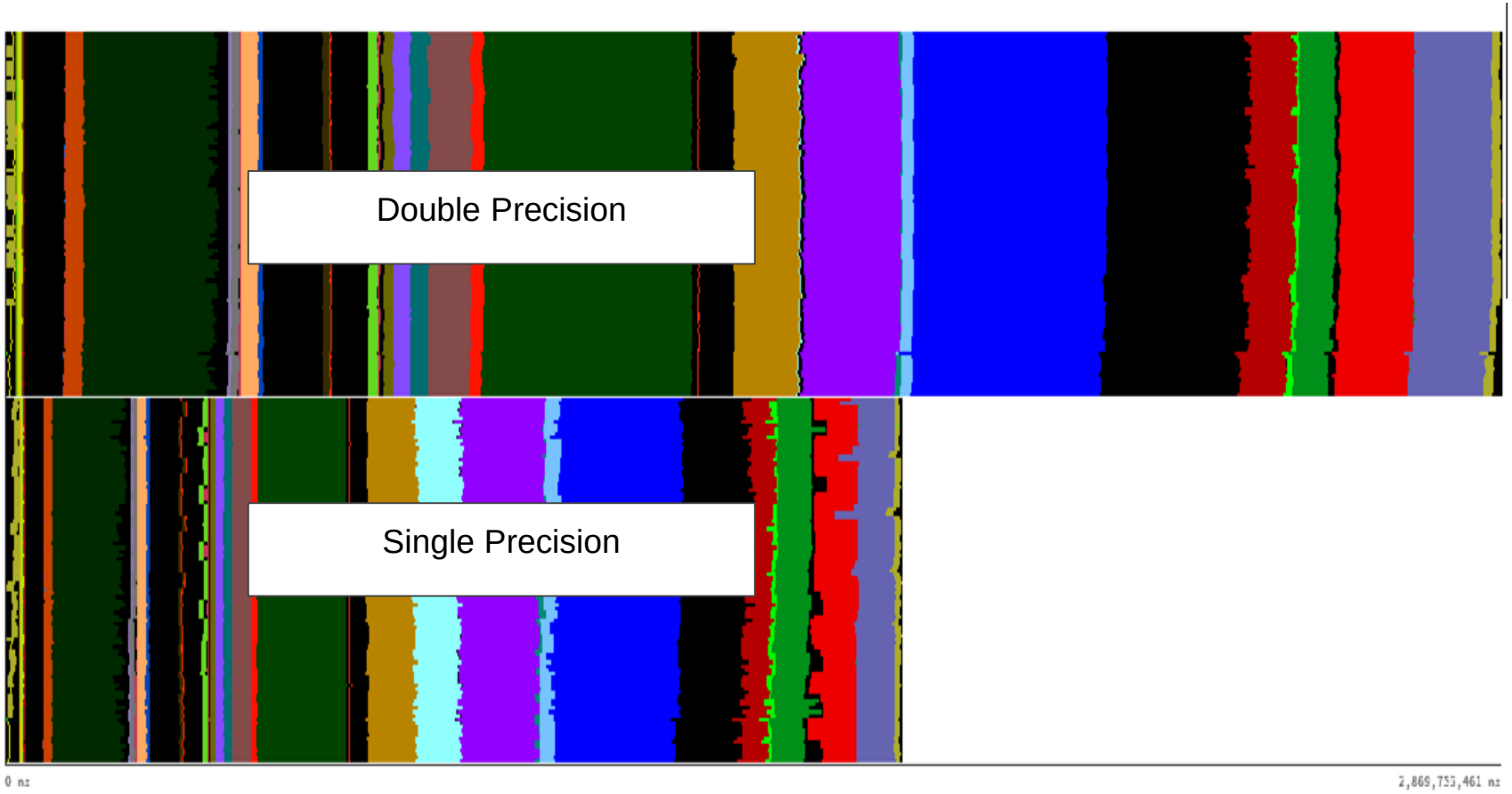
96 processes

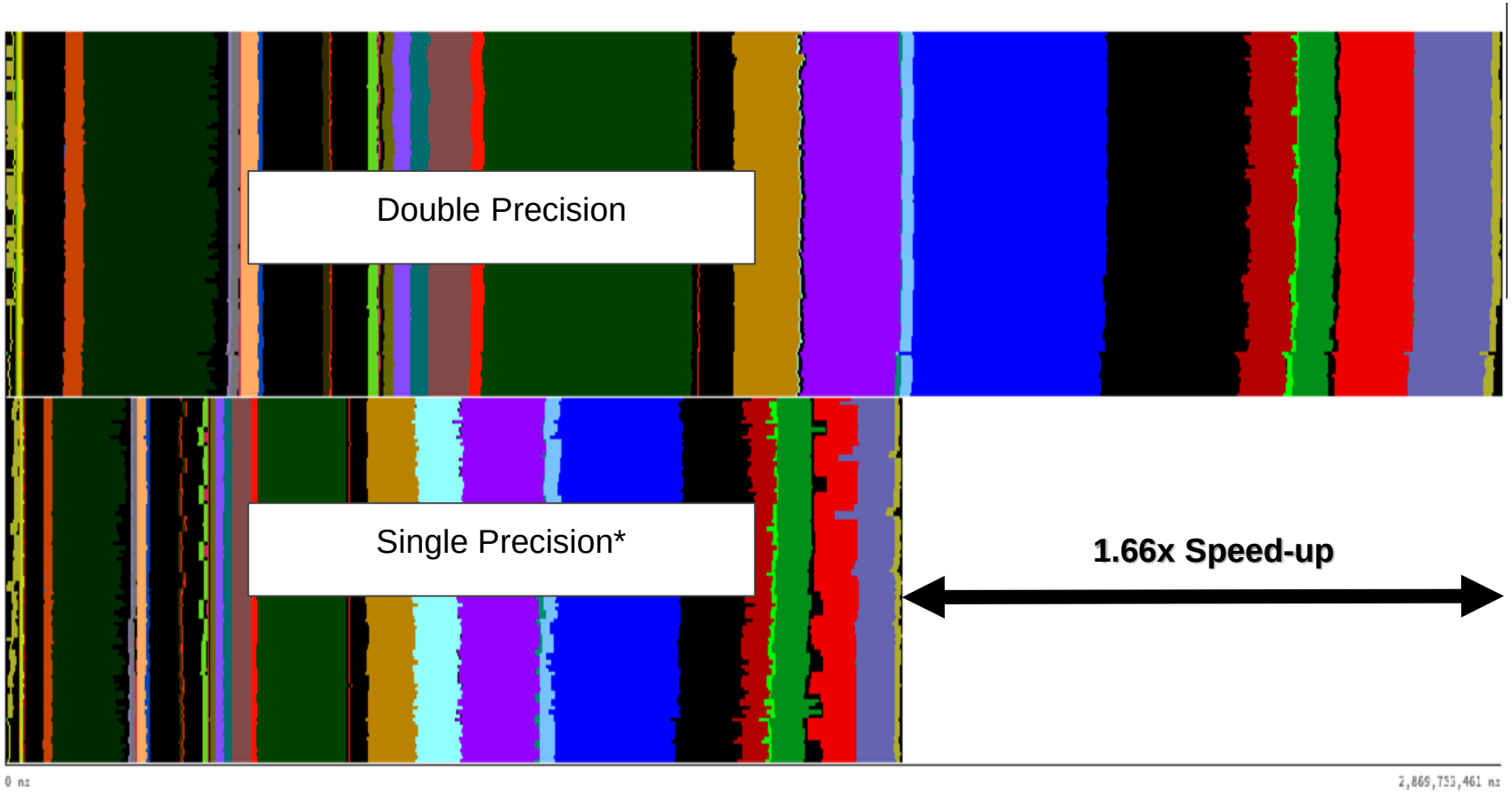


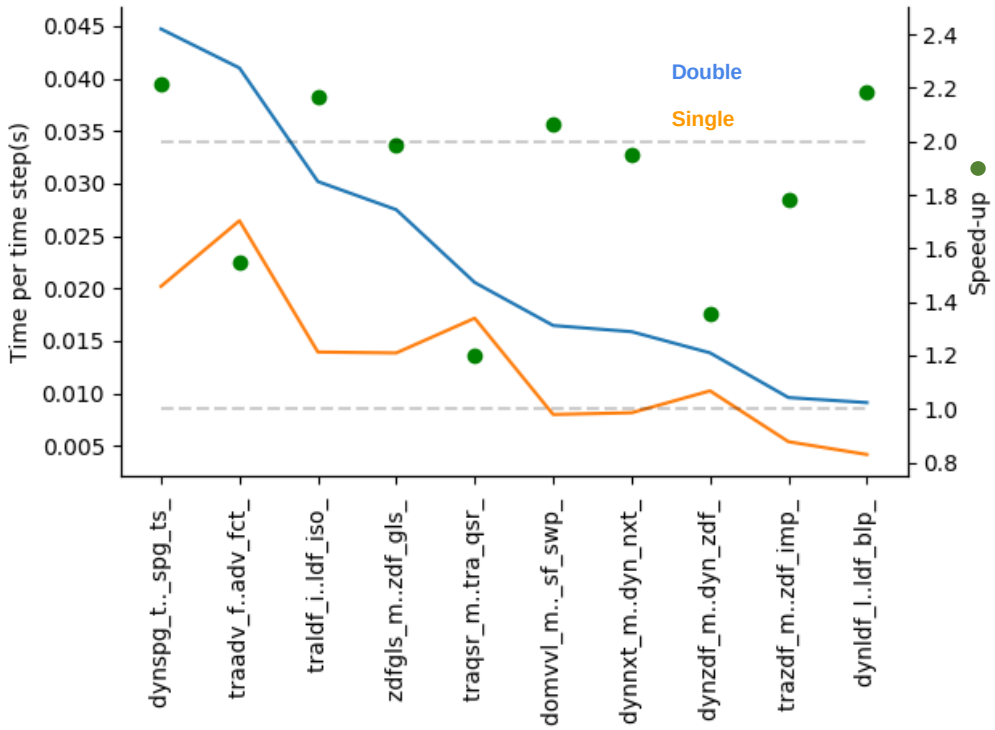
Time

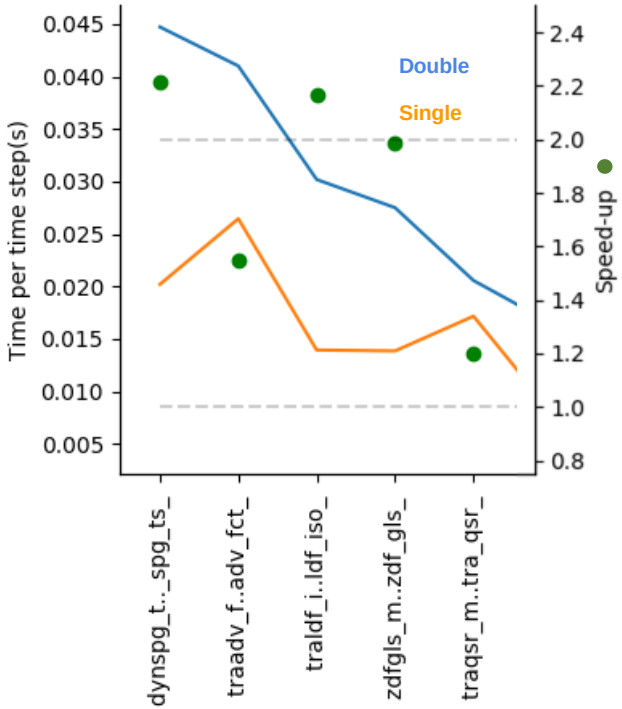


- The trace corresponds to a single time-step
- NEMO 4.0 Ocean-only
- ORCA025 -> global ¼ º ~ 27km at equator
- Each color represents a different routine

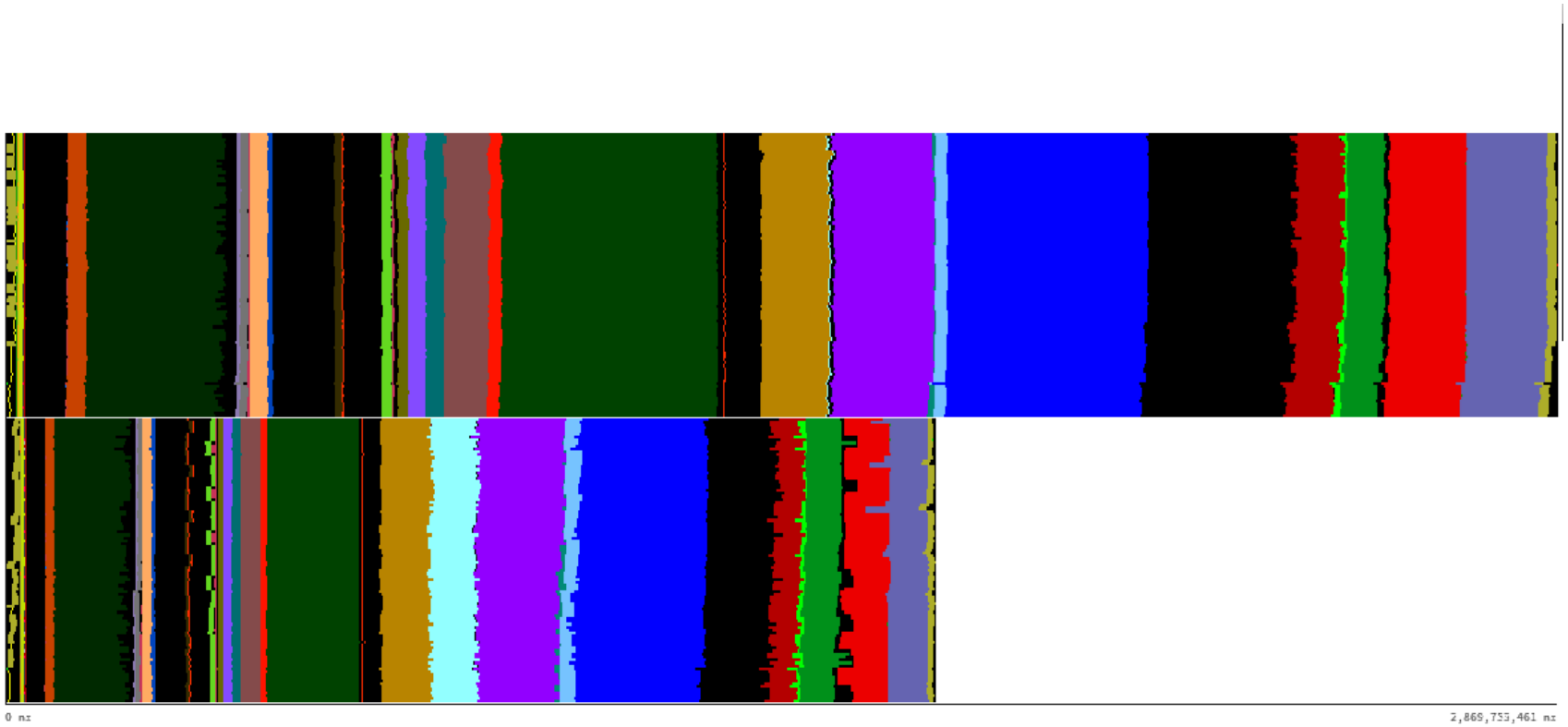

Double Precision
Single Precision
0 nz
2,869,753,461 nz


Double Precision
Single Precision*
1.66x Speed-up
0 nz
2,869,753,461 nz

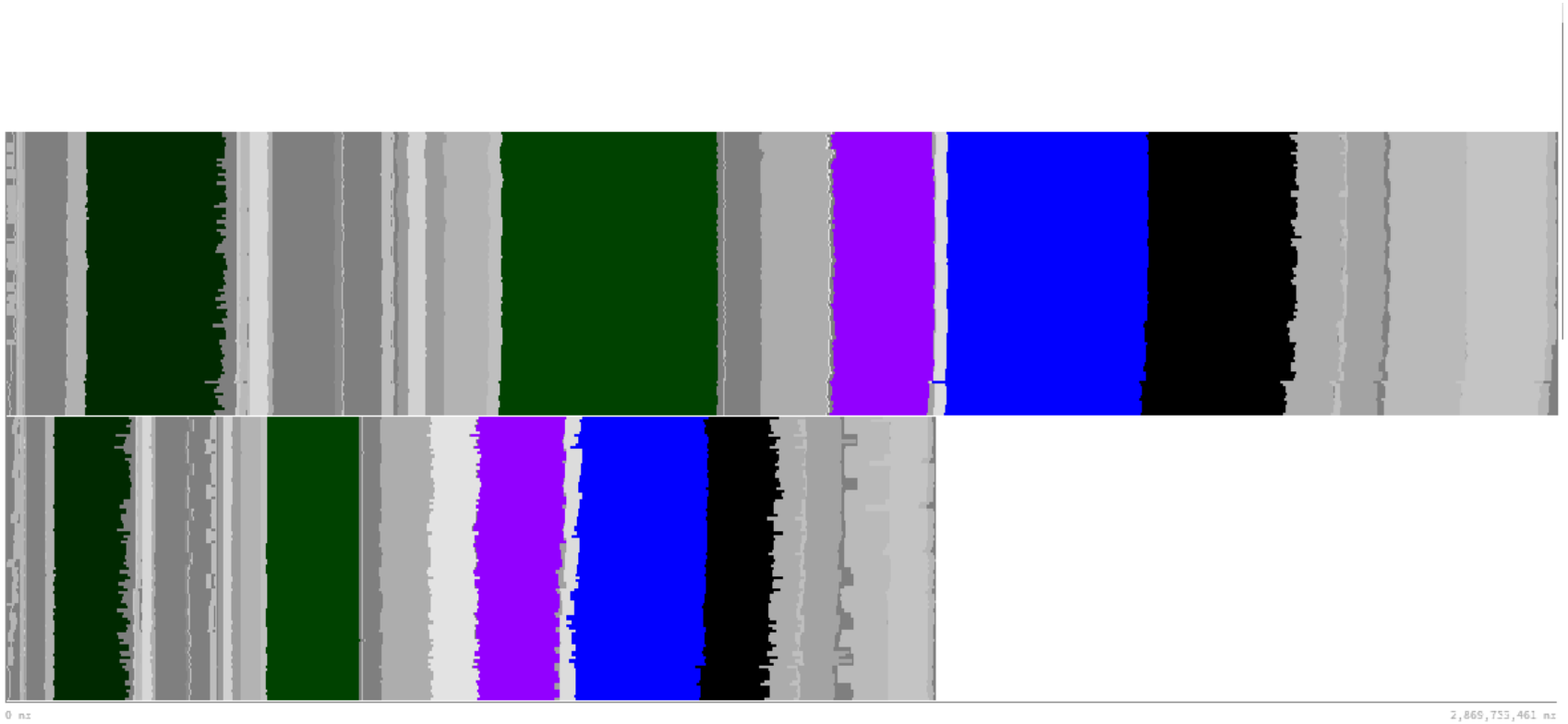

Time per time step(s)
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
Speed-up
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
Double
Single
dynspg_t..._spg_ts_
traadv_f..adv_fct_
traldf_i..ldf_iso_
zdfgls_m..zdf_gls_
traqsr_m..tra_qsr_
domvvl_m..._sf_swp_
dynnxt_m..dyn_nxt_
dynzdf_m..dyn_zdf_
trazdf_m..zdf_imp_
dynldf_l..ldf_blp_


Time per time step(s)
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
Speed-up
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
Double
Single
dynspg_t..._spg_ts_
traadv_f..adv_fct_
traldf_i..ldf_iso_
zdfgls_m..zdf_gls_
traqsr_m..tra_qsr_

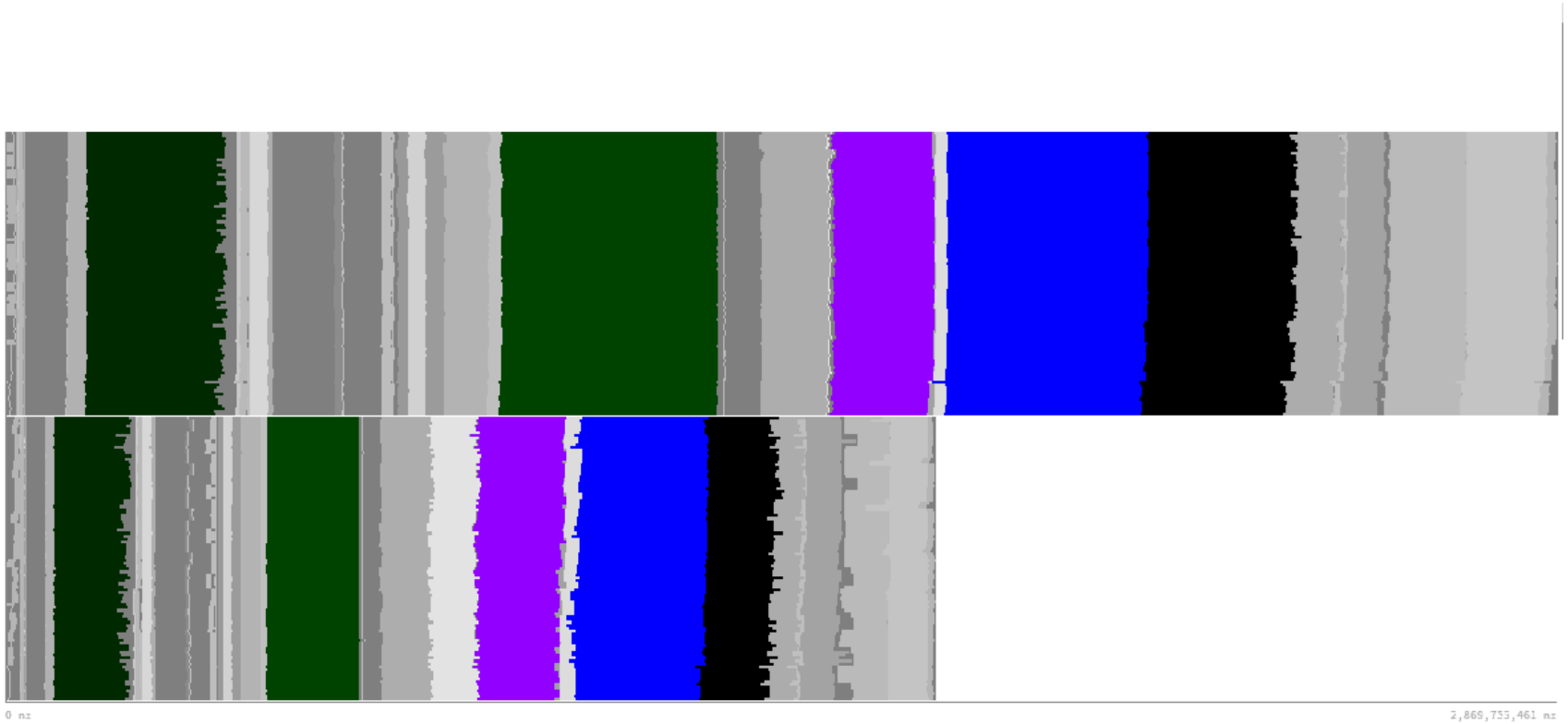
dynspg traadv traldf zdfgls traqsr



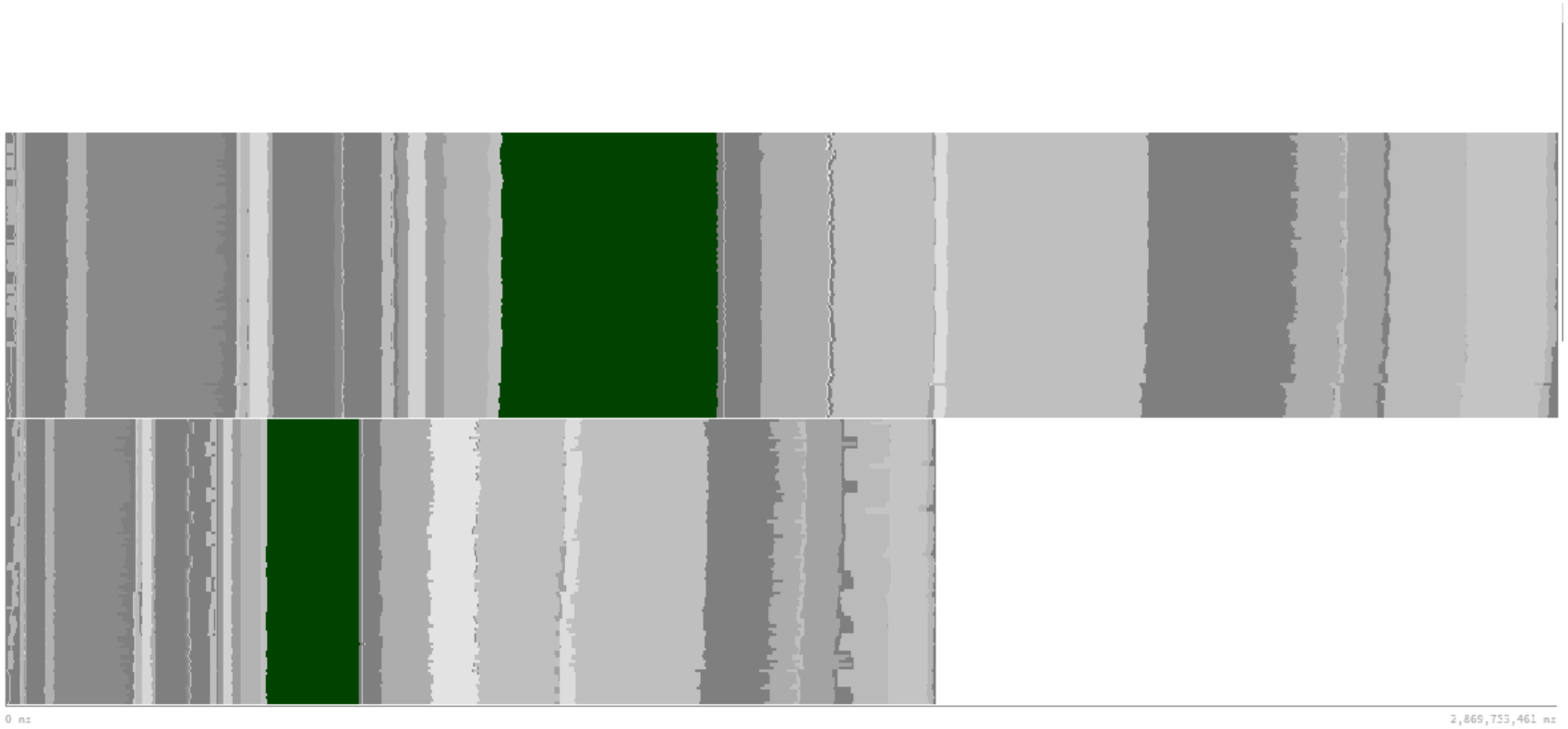
dynspg traadv traldf zdfgls traqsr



dynspg traadv traldf zdfgls traqsr

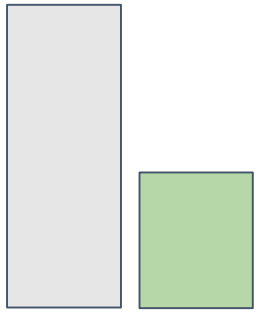


dynspg traadv traldf zdfgls traqsr



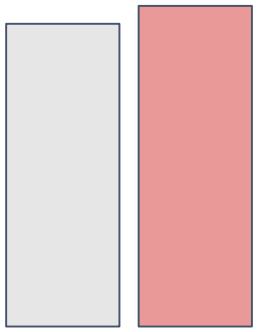
dynspg traadv traldf zdfgls traqsr

Iteration time

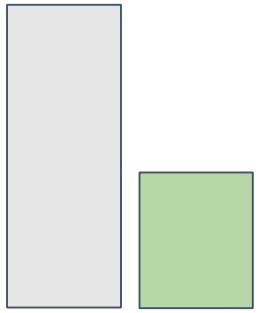


dynspg traadv traldf zdfgls traqsr

Total Instructions



Iteration time

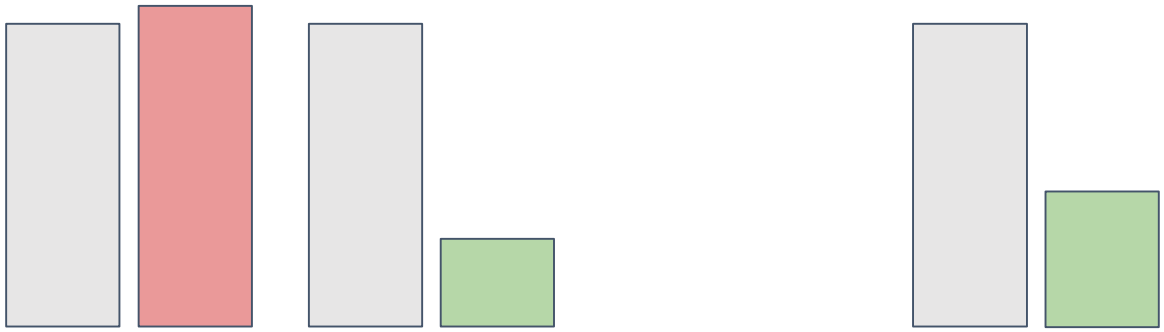


dynspg traadv traldf zdfgls traqsr

Total Instructions

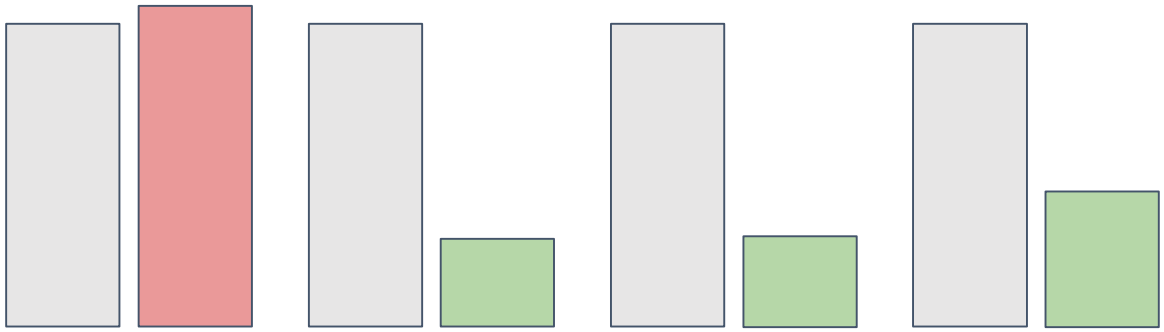
Cycles per Instruction

Iteration time

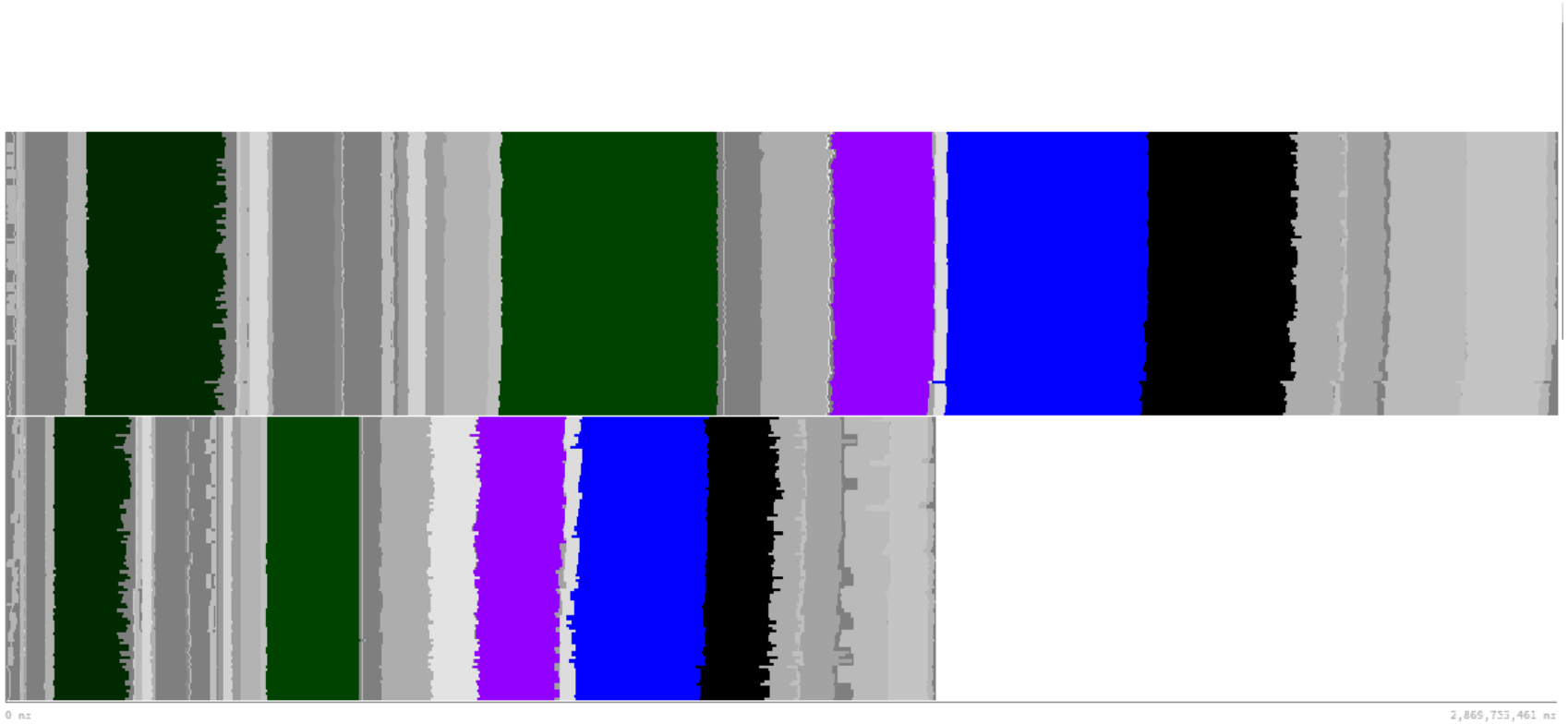


dynspg traadv traldf zdfgls traqsr

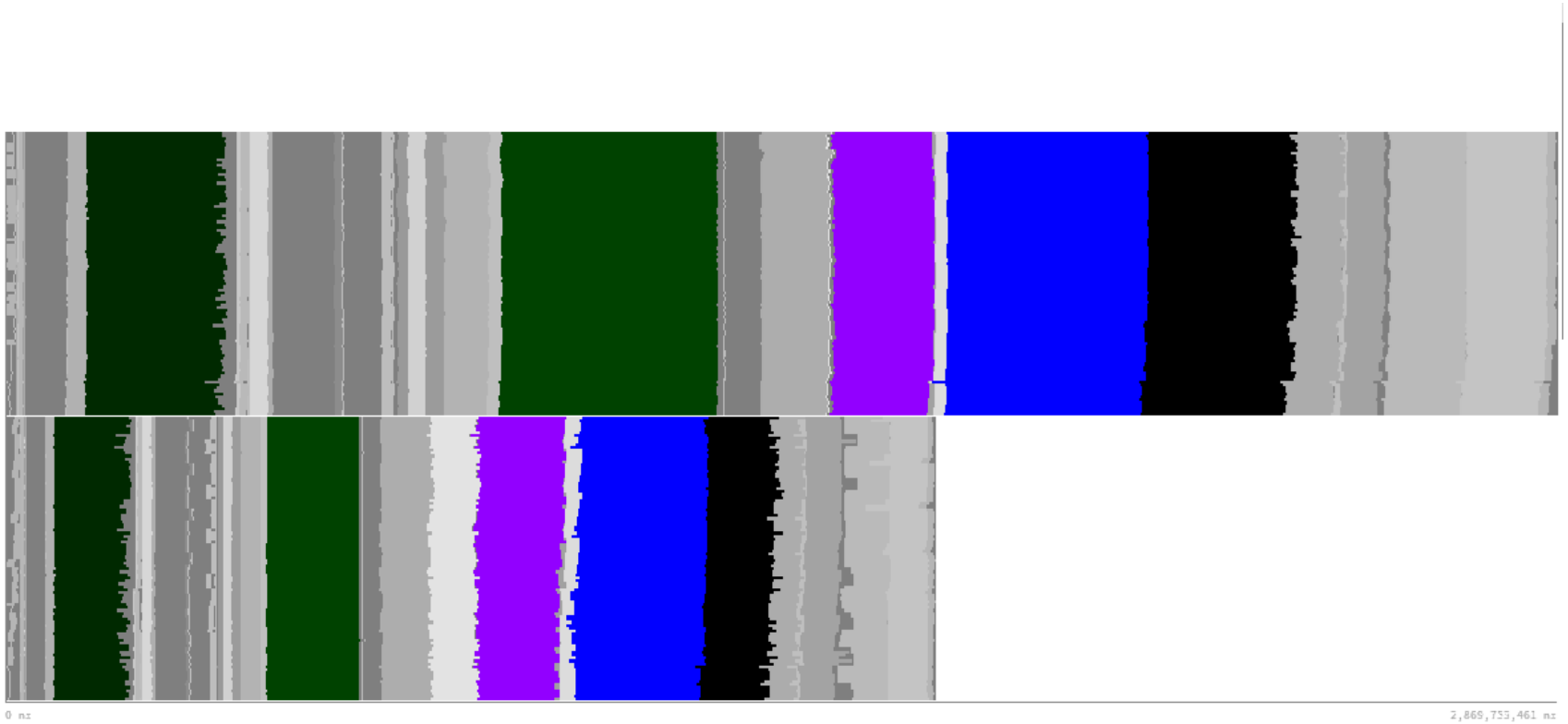
Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time



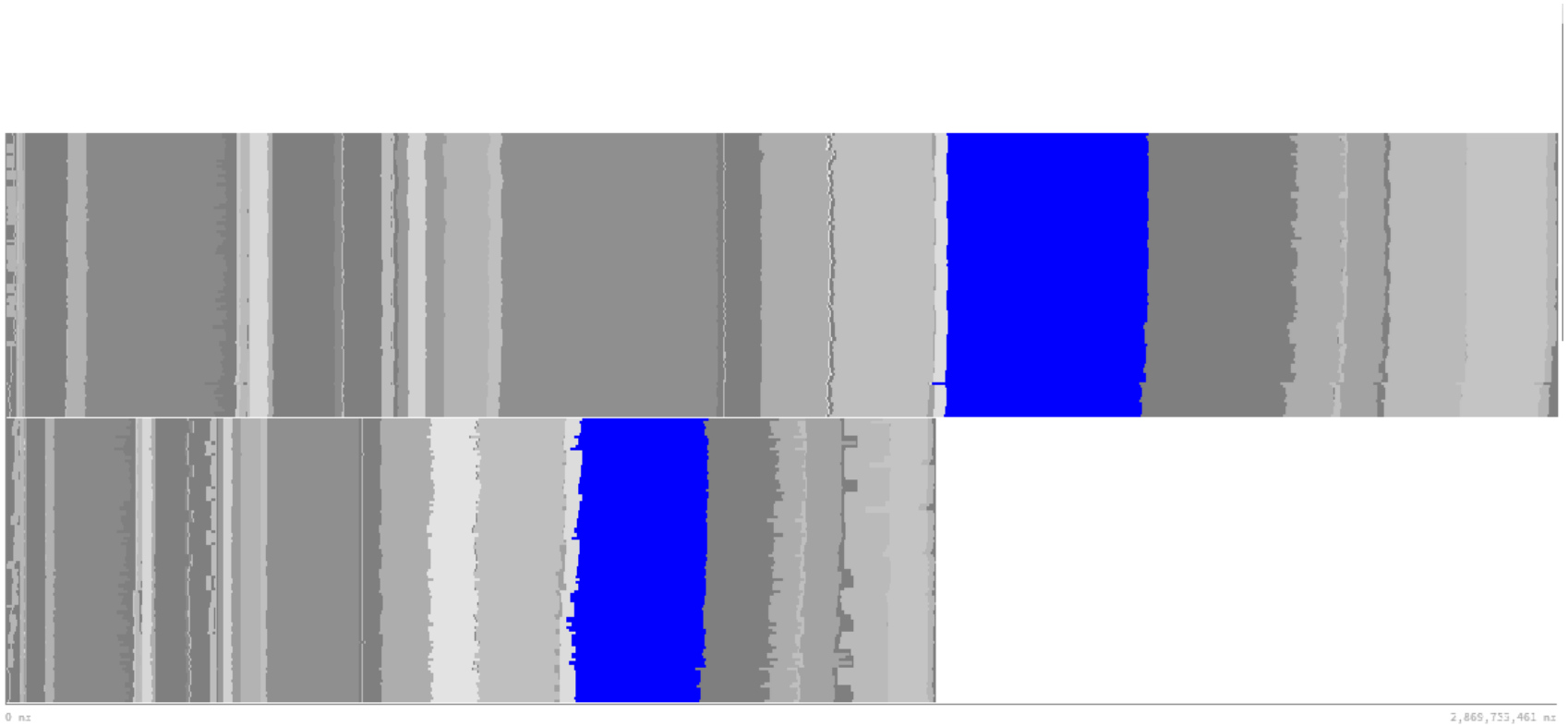
dynspg traadv traldf zdfgls traqsr



dynspg **traadv** traldf zdfgls traqsr

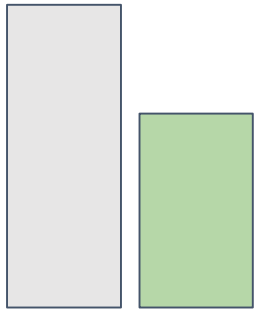


dynspg **traadv** traldf zdfgls traqsr



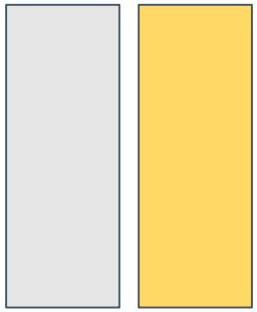
dynspg **traadv** traldf zdfgls traqsr

Iteration time

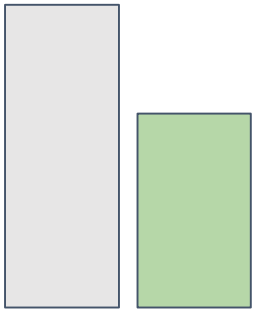


dynspg **traadv** traldf zdfgls traqsr

Total Instructions



Iteration time

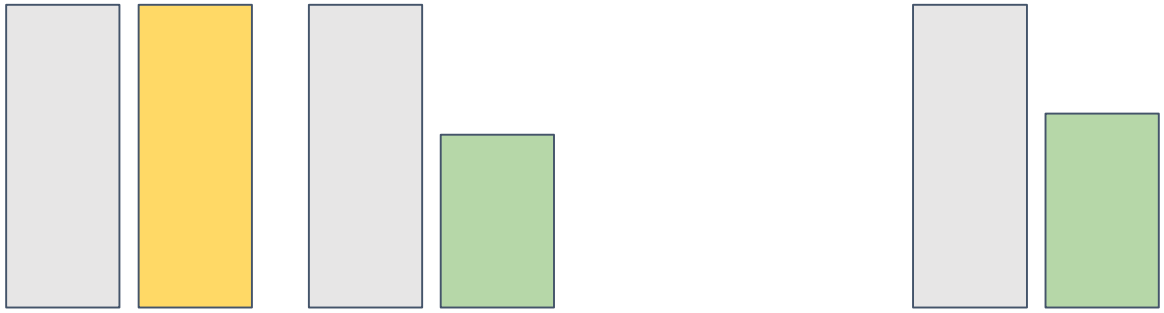


dynspg **traadv** traldf zdfgls traqsr

Total Instructions

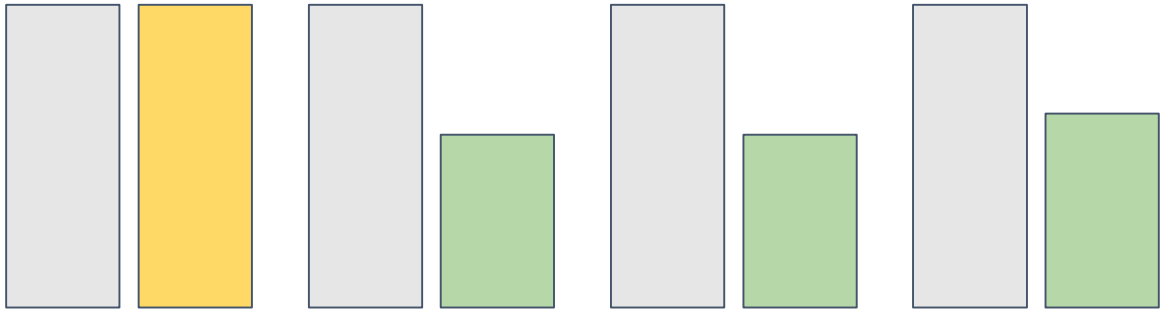
Cycles per Instruction

Iteration time

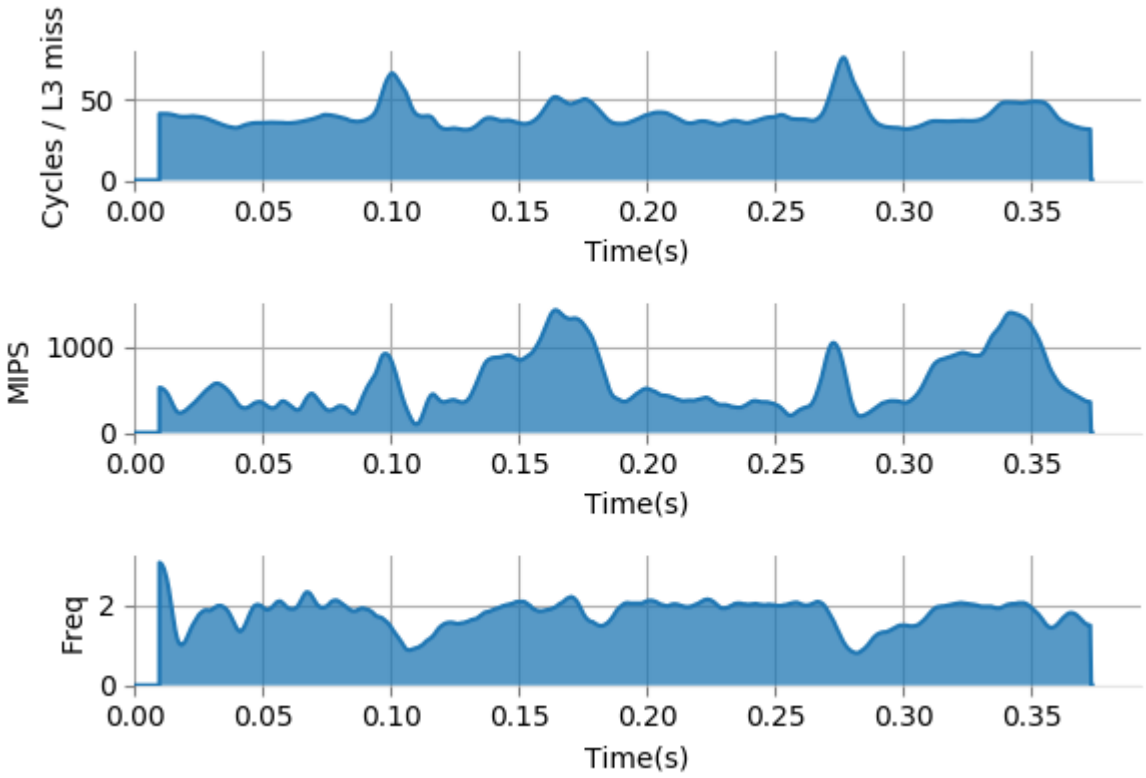


dynspg **traadv** traldf zdfgls traqsr

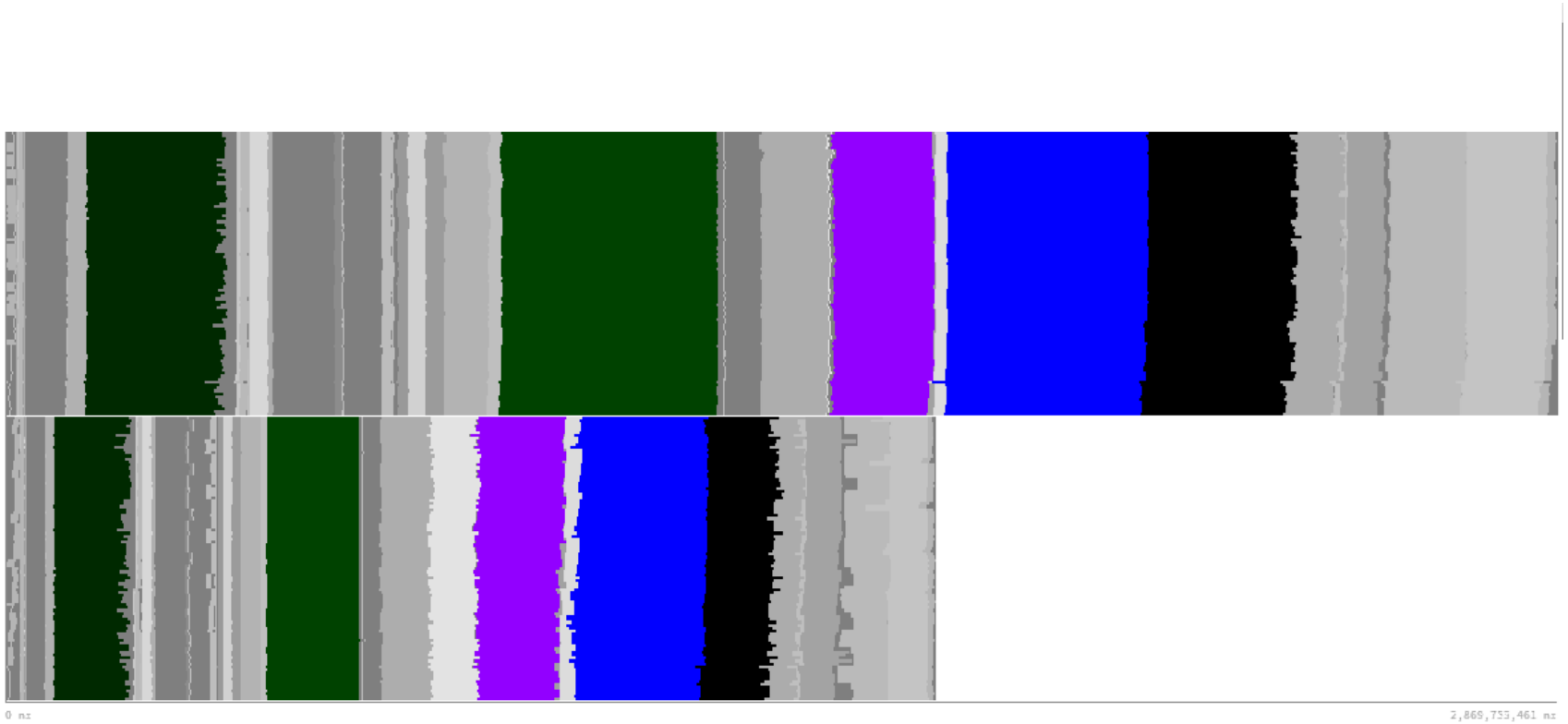
Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time



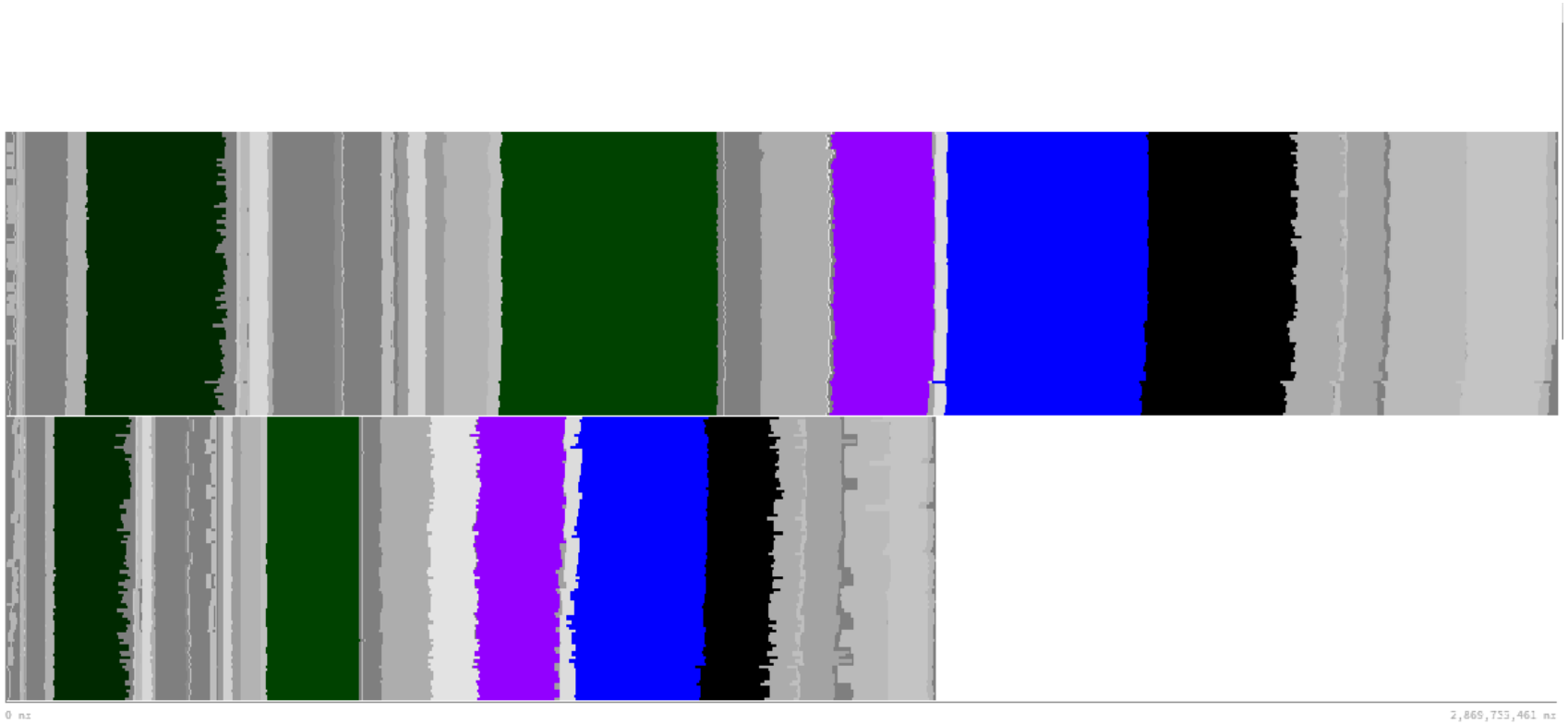
dynspg **traadv** traldf zdfgls traqsr



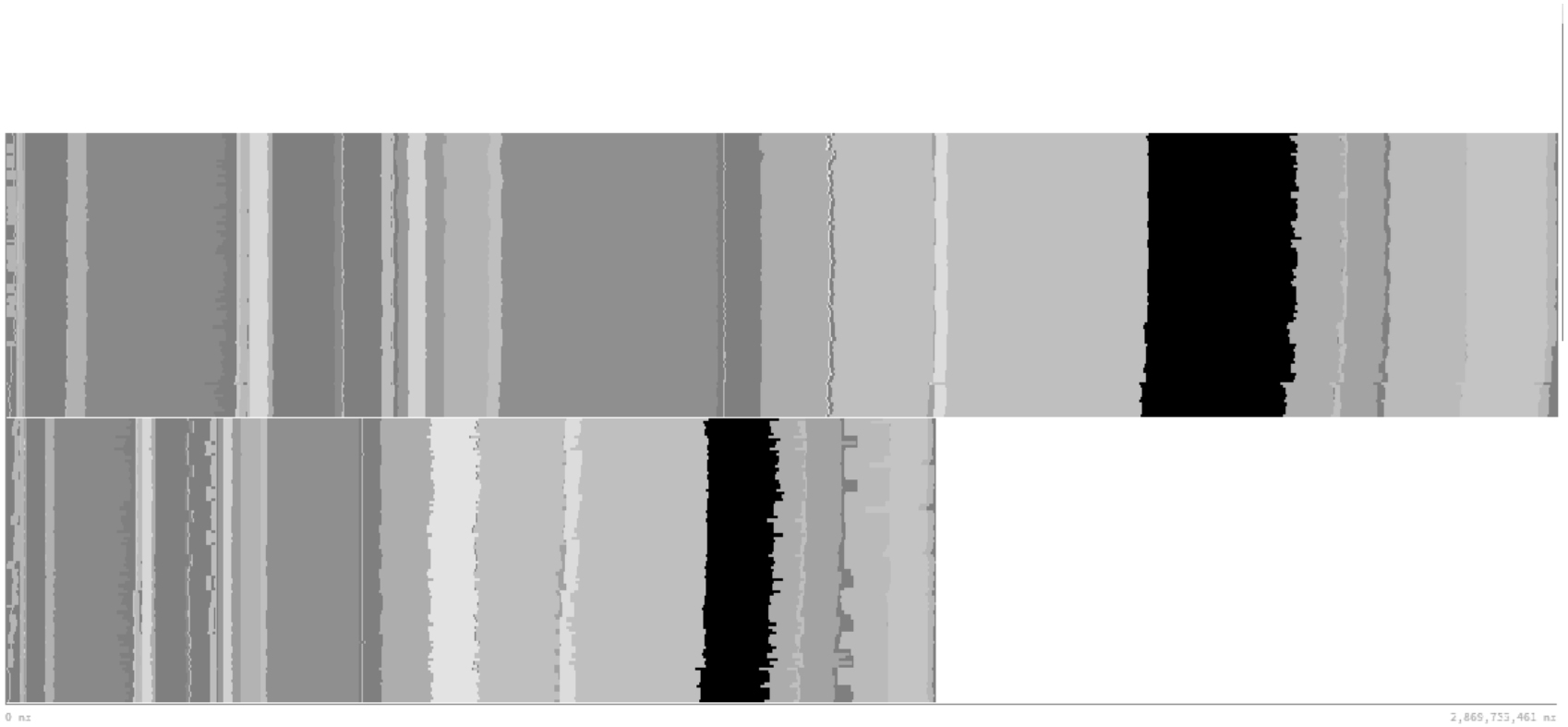
dynspg traadv traldf zdfgls traqsr



dynspg traadv **traldf** zdfgls traqsr

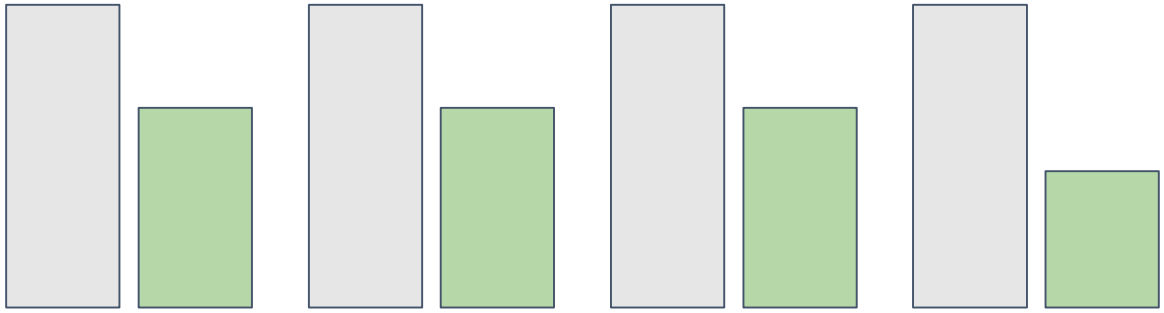


dynspg traadv **traldf** zdfgls traqsr

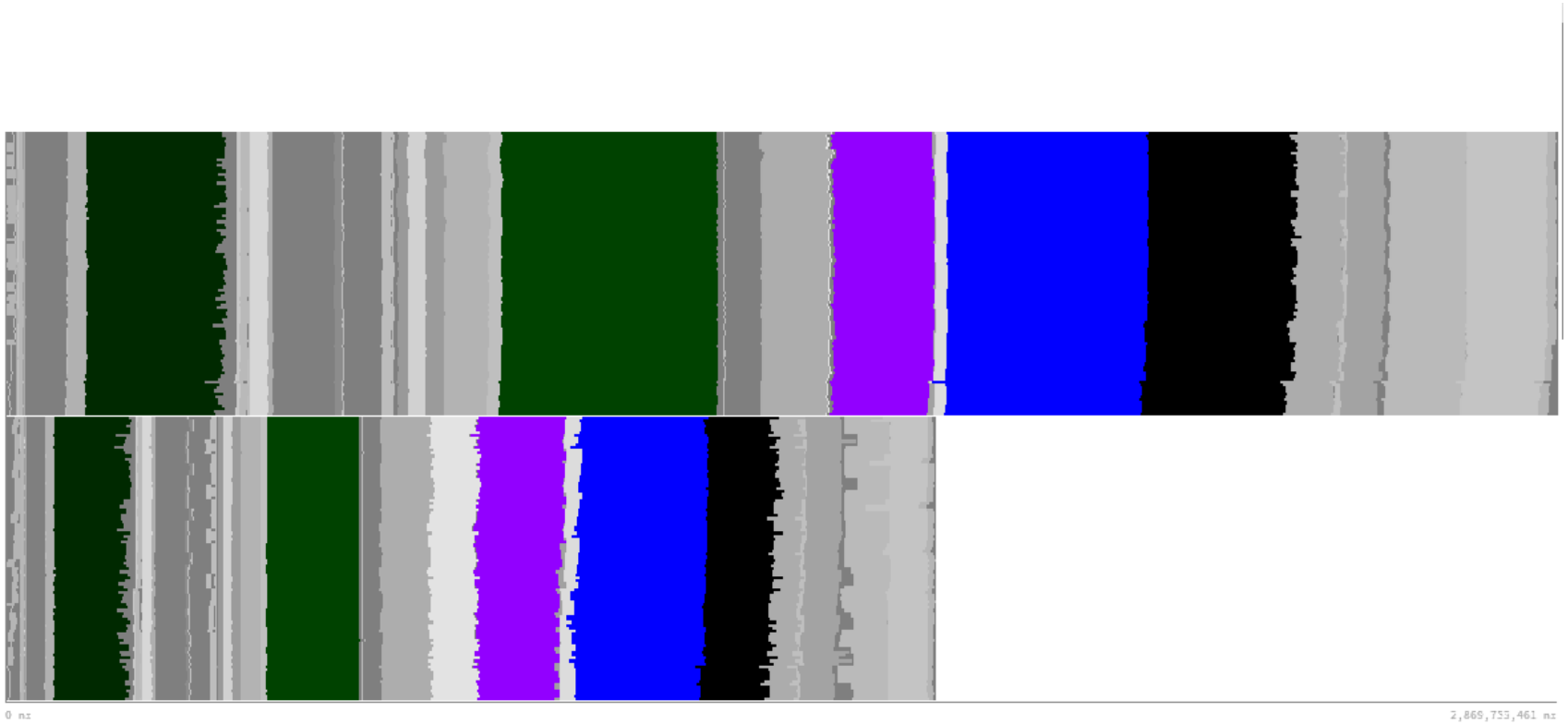


dynspg traadv **traldf** zdfgls traqsr

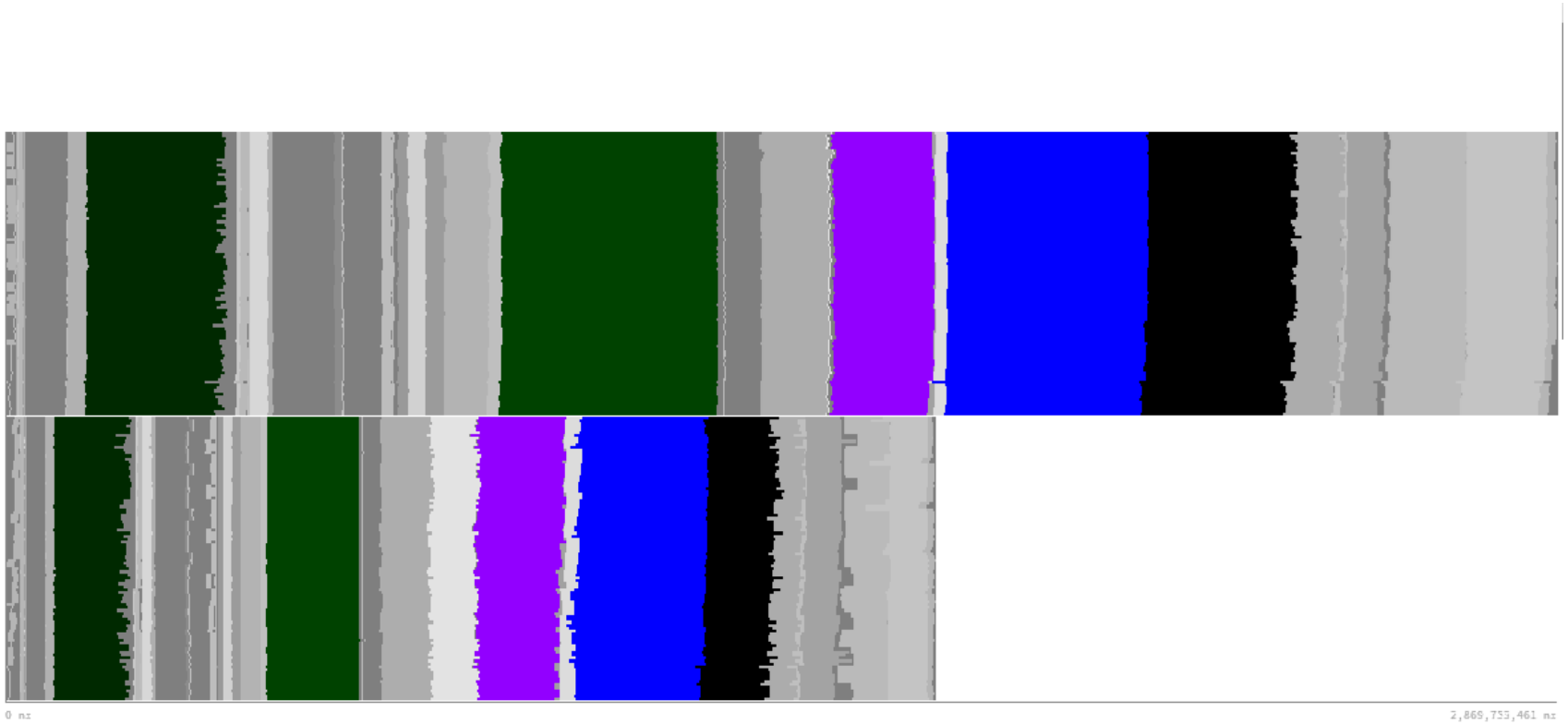
Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time



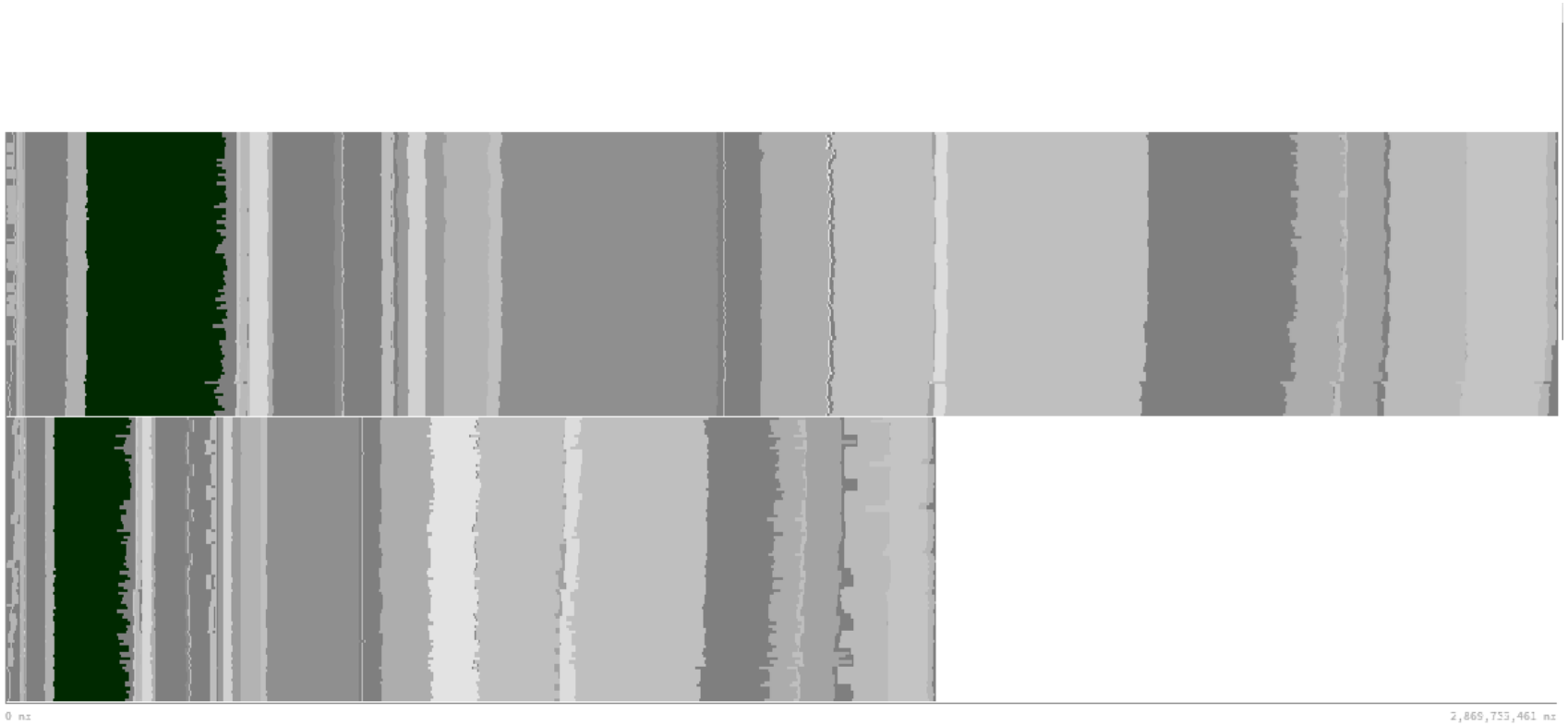
dynspg traadv traldf zdfgls traqsr



dynspg traadv traldf **zdfgls** traqsr

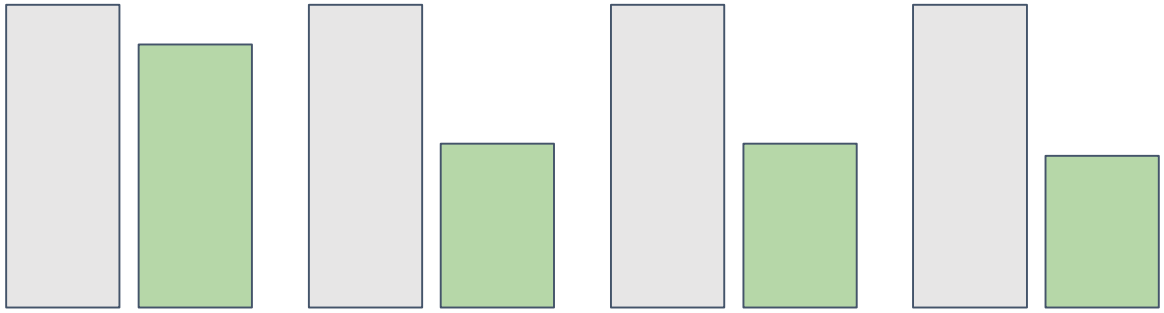


dynspg traadv traldf **zdfgls** traqsr

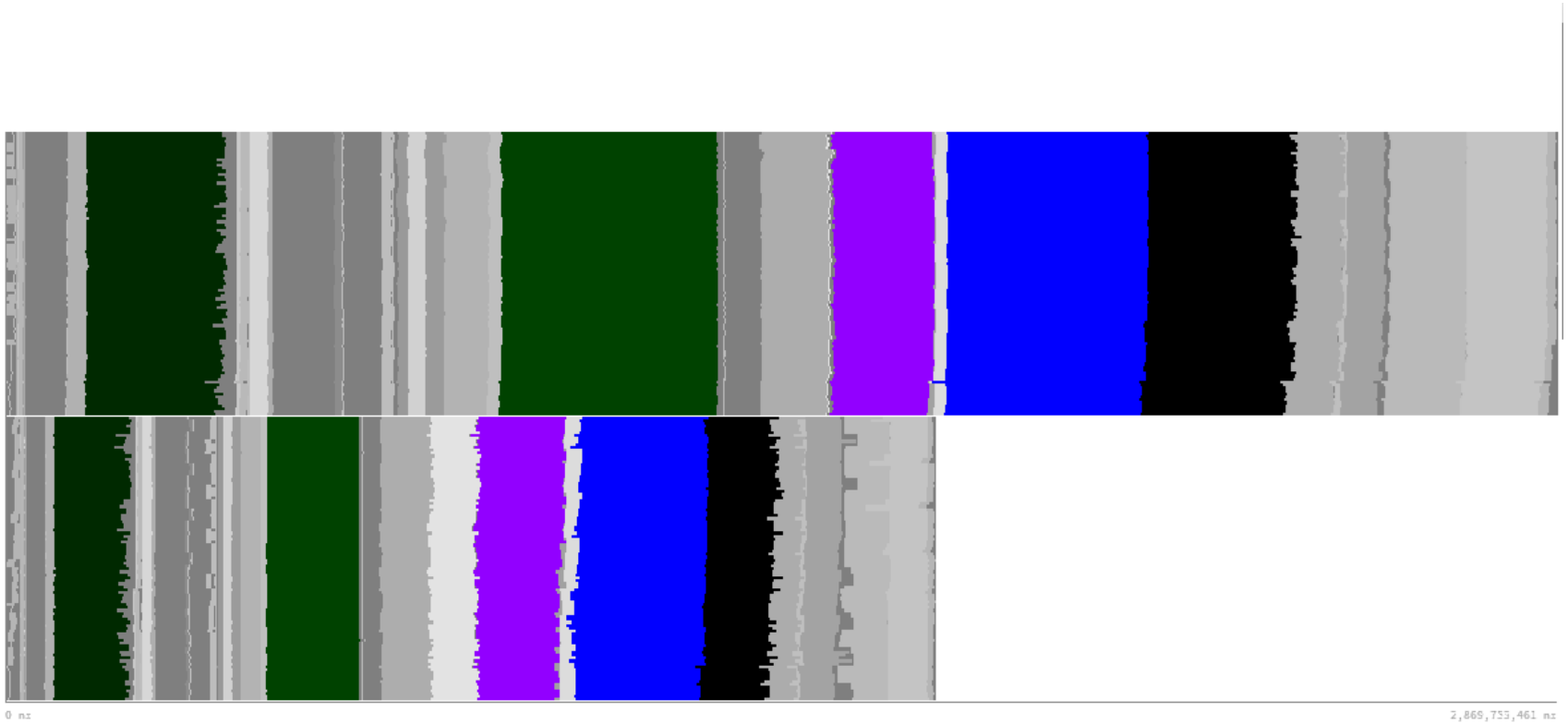


dynspg traadv traldf **zdfgls** traqsr

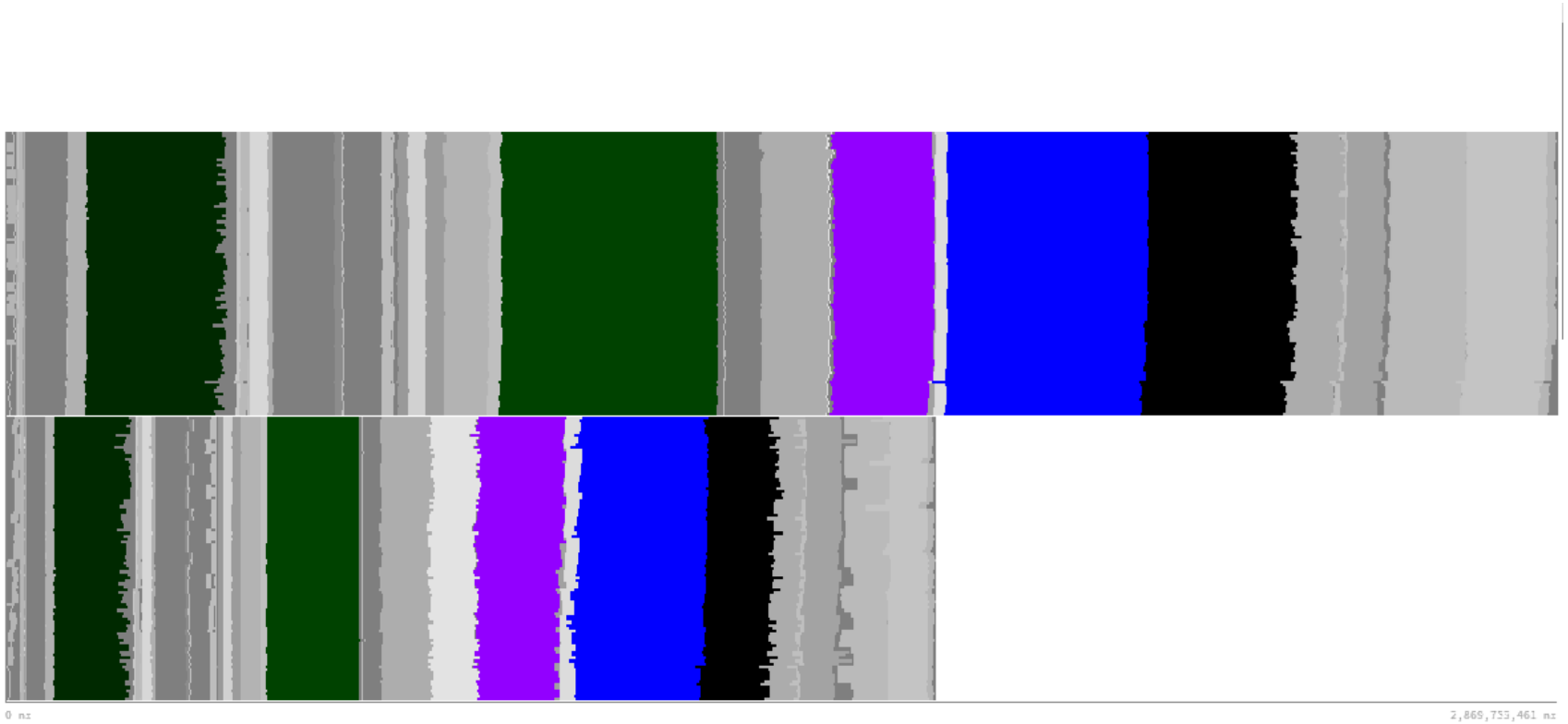
Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time



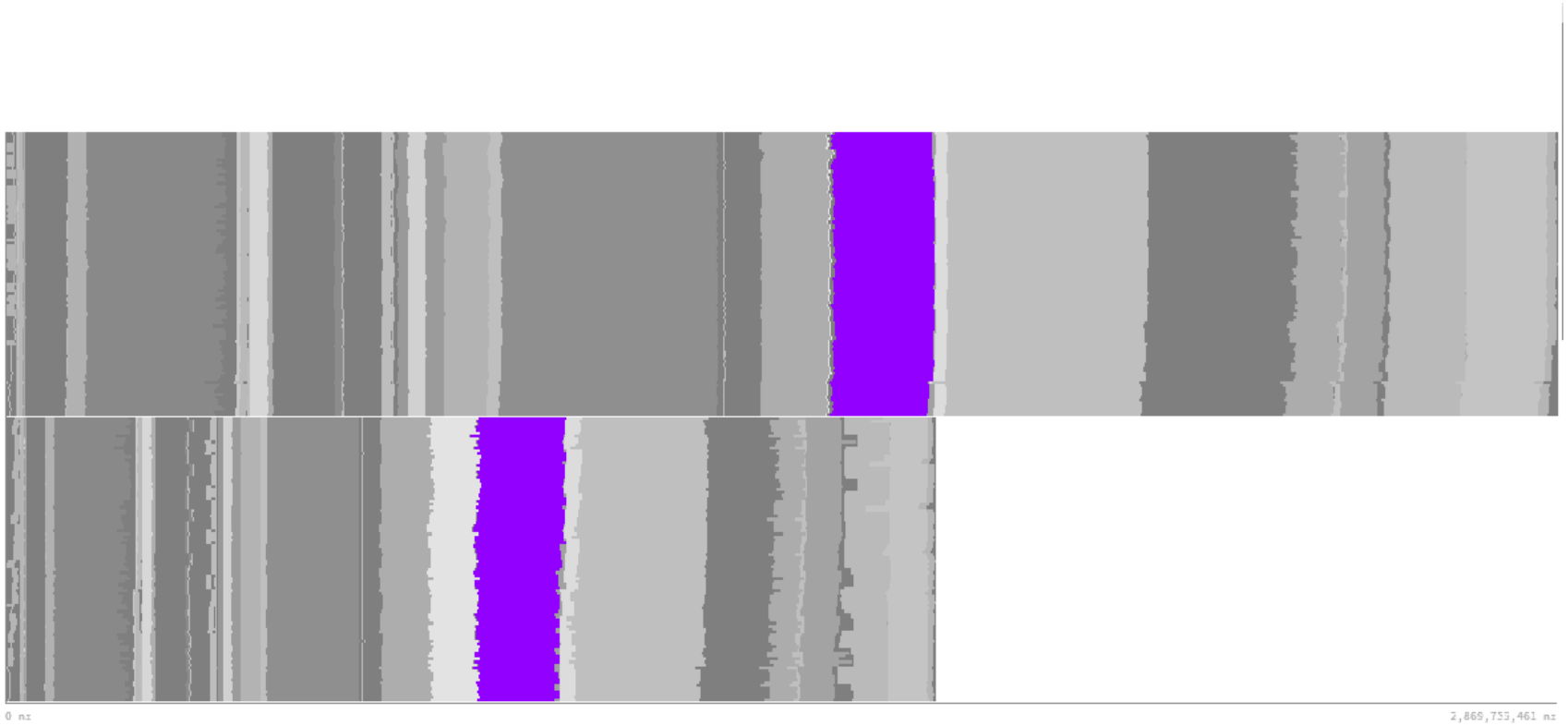
dynspg traadv traldf zdfgls traqsr



dynspg traadv traldf zdfgls **traqsr**



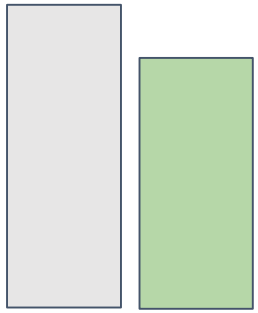
dynspg traadv traldf zdfgls **traqsr**



dynspg traadv traldf zdfgls **<u>traqsr</u>**

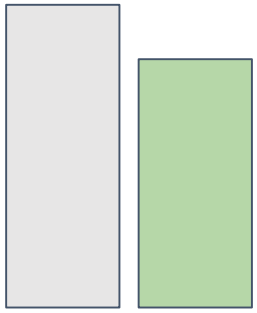
Speed-up **1.2x**

Iteration time

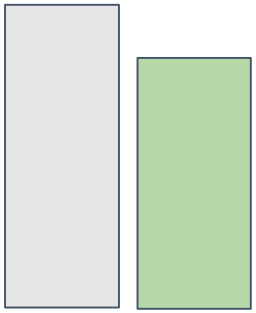


dynspg traadv traldf zdfgls **<u>traqsr</u>**

Total Instructions



Iteration time

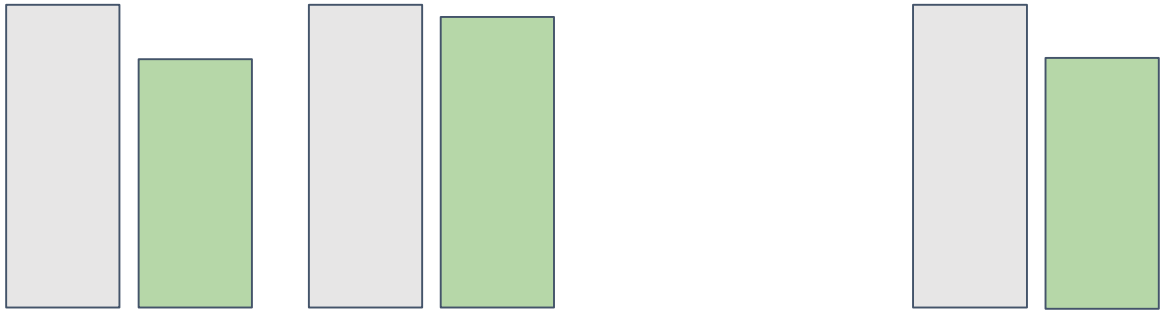


dynspg traadv traldf zdfgls **traqsr**

Total Instructions

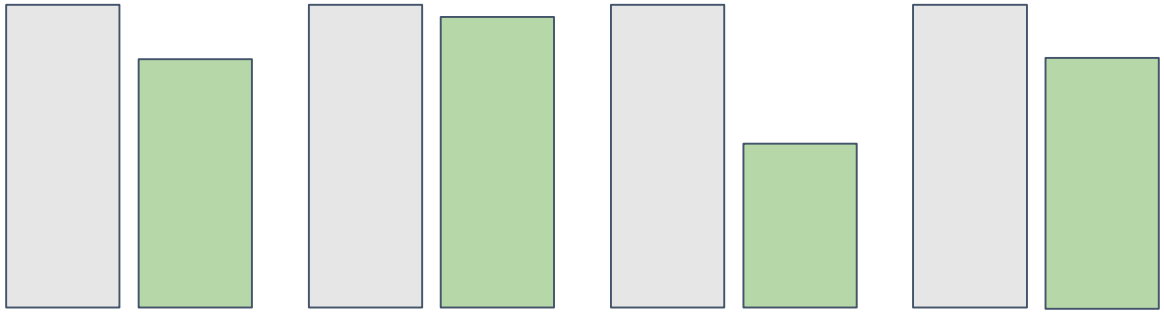
Cycles per Instruction

Iteration time



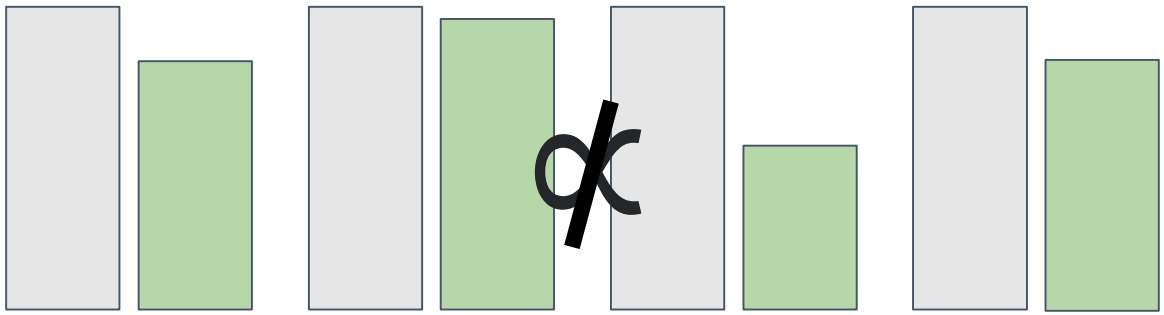
dynspg traadv traldf zdfgls **traqsr**

Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time

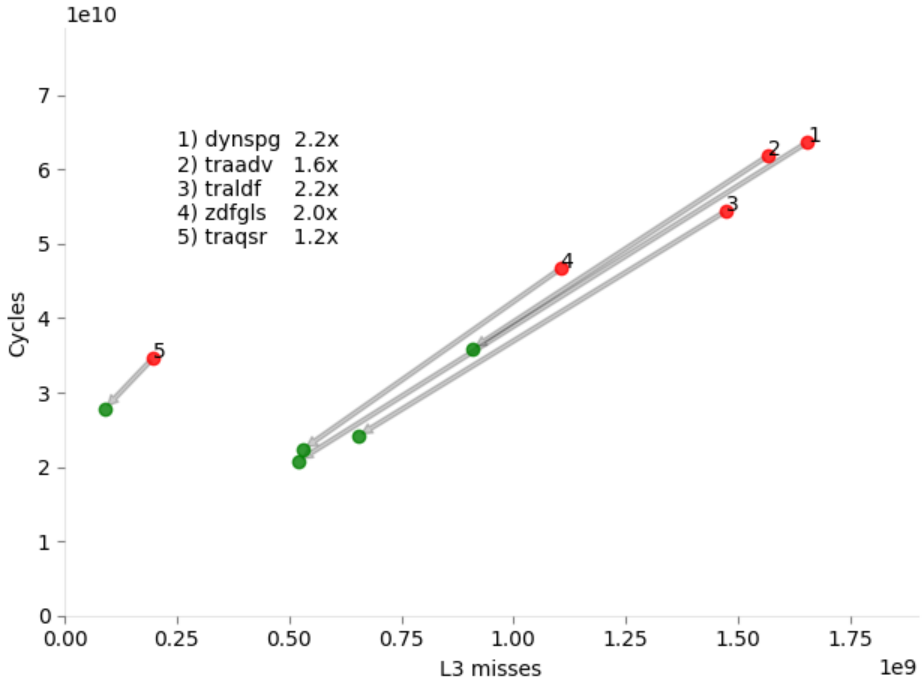


dynspg traadv traldf zdfgls **traqsr**

Total Instructions | Cycles per Instruction | L3 miss per Instruction | Iteration time

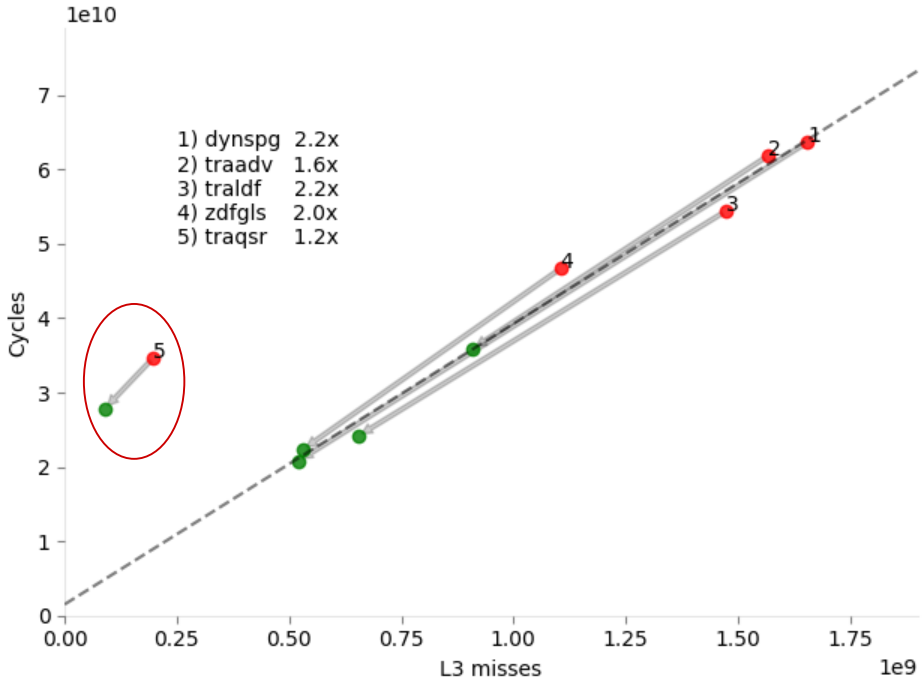


What happens with **traqsr?**

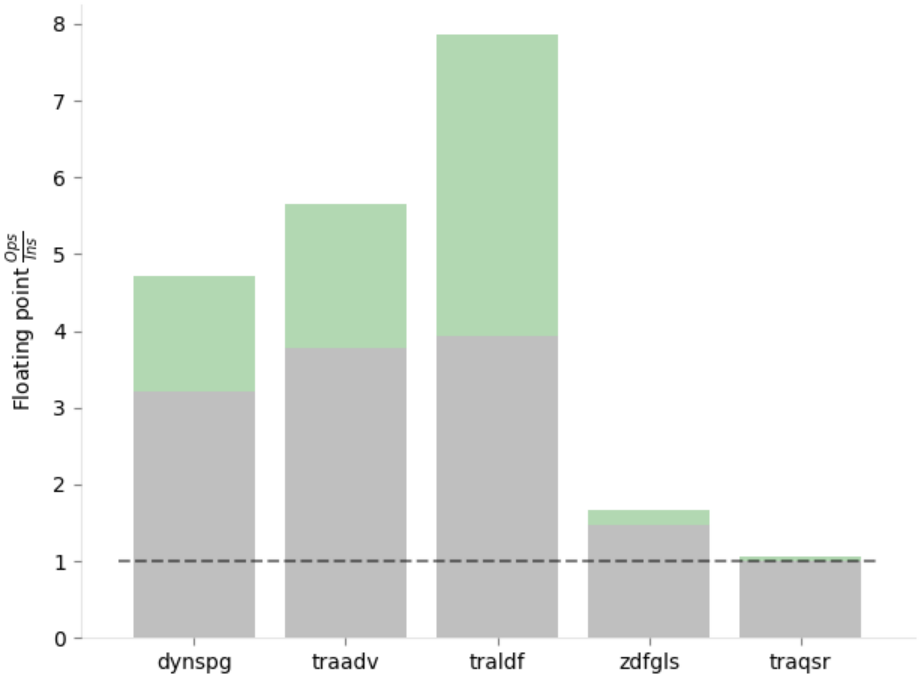


What happens with **traqsr?**

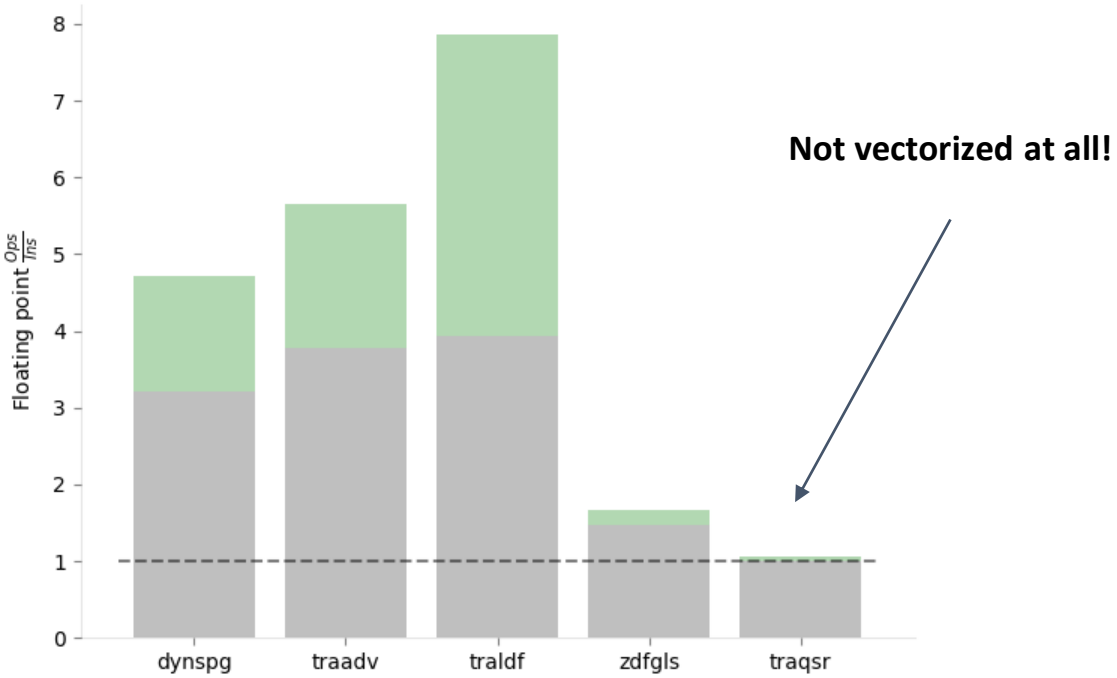
r2=0.99



What happens with **traqsr?**

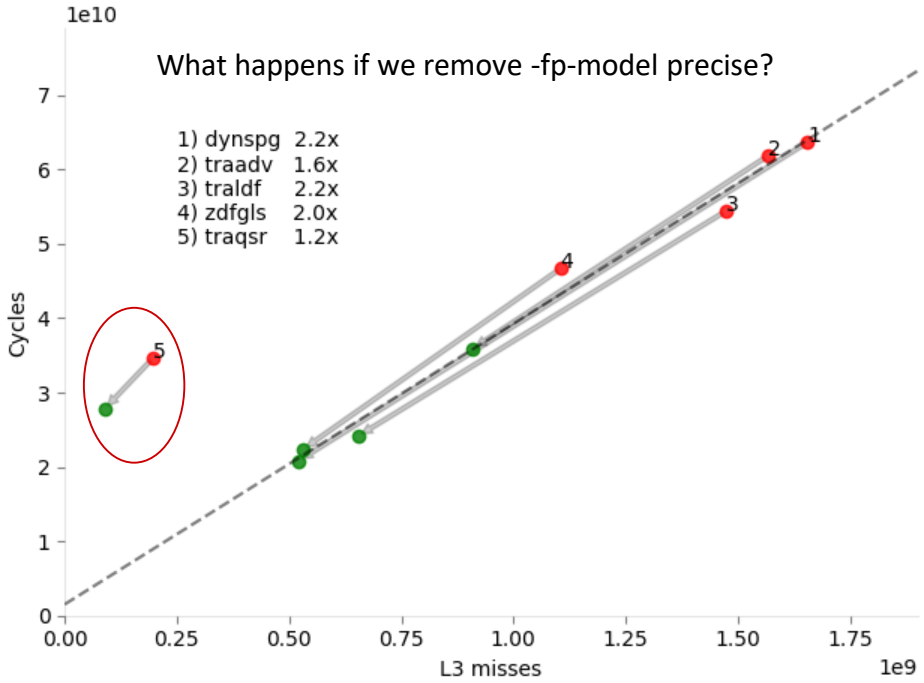


What happens with **traqsr?**



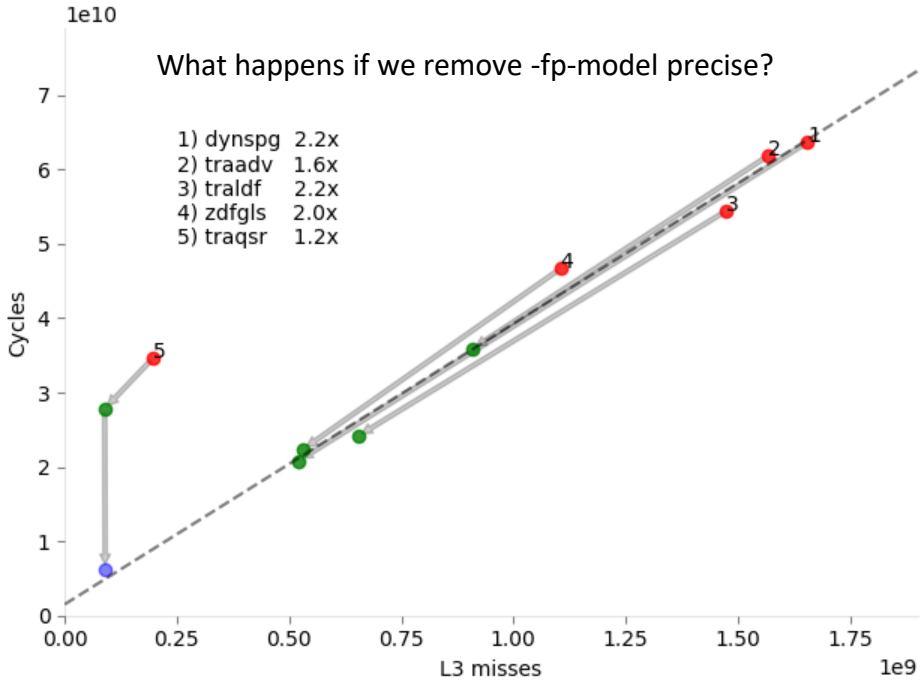
What happens with **traqsr?**

r2=0.99

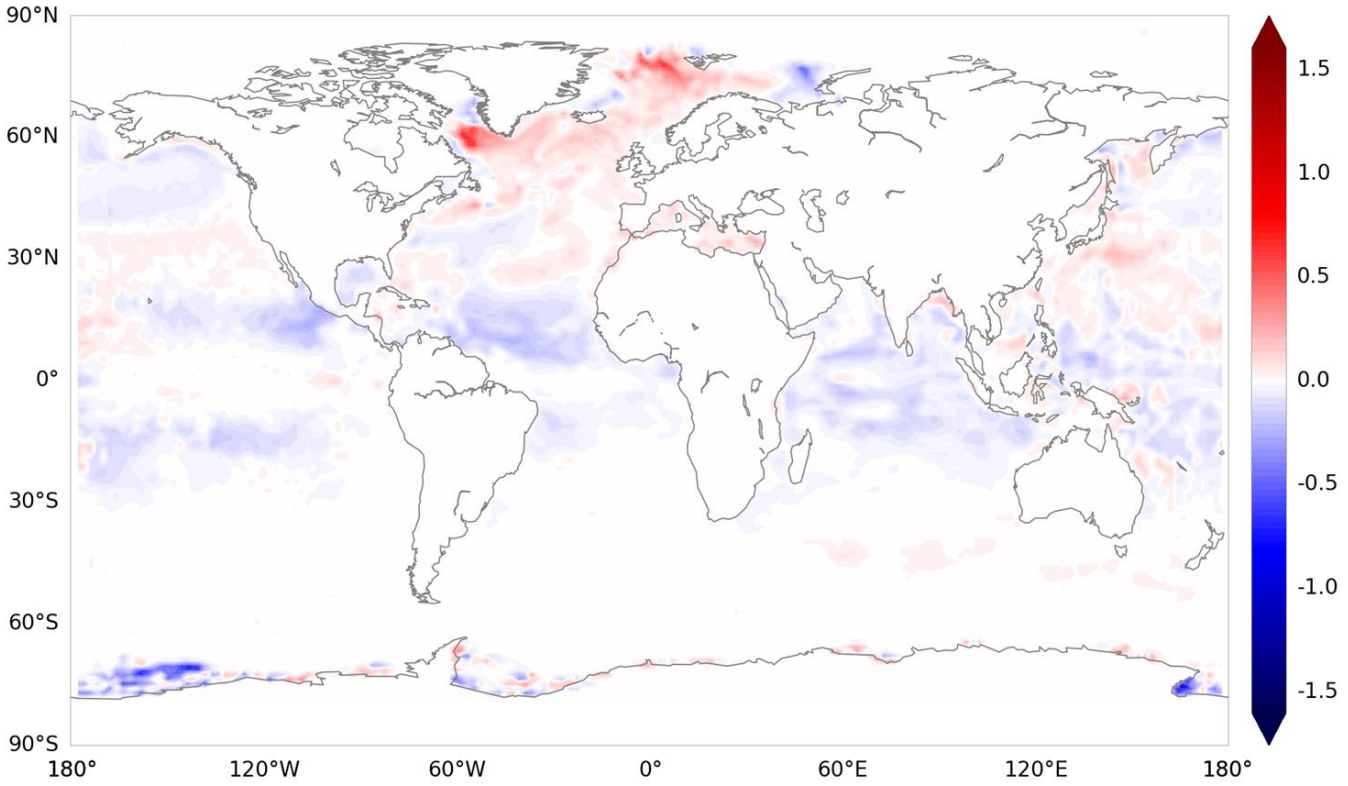


What happens with **traqsr?**

r2=0.99

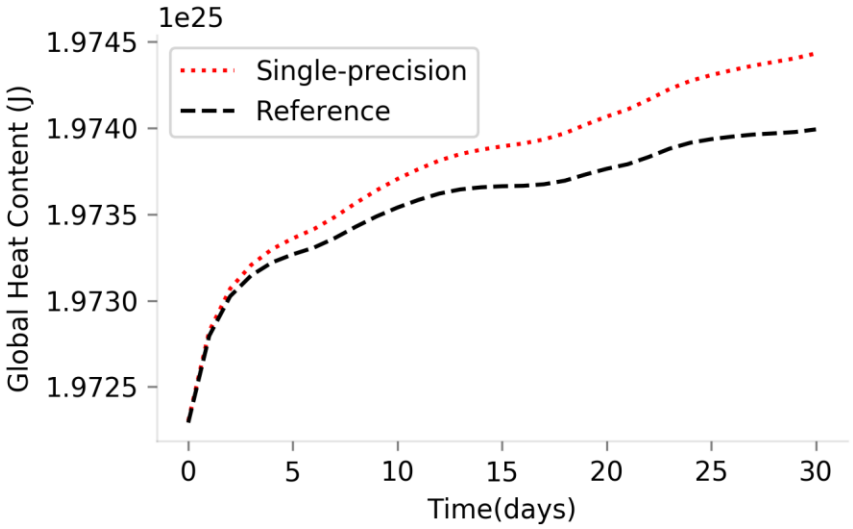


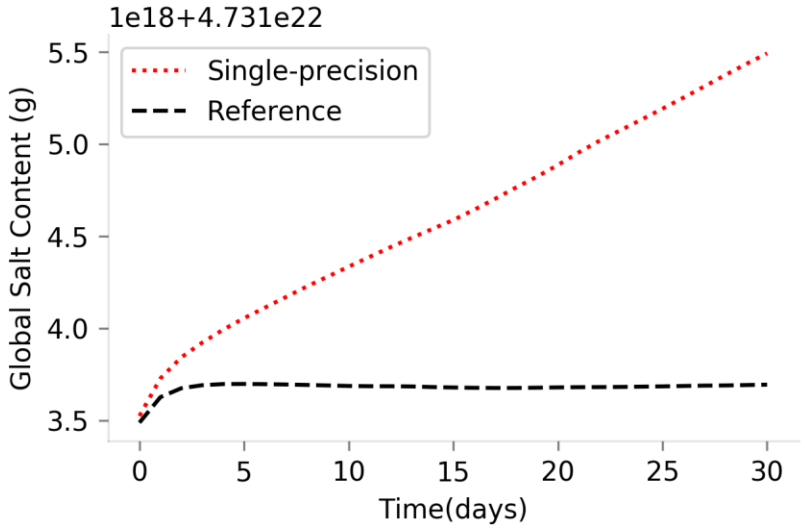
But what about the results?



Difference between double- and single-precision monthly mean of sea-surface temperature for the first month of simulation.

But what about the results?





Risks of low precision

Solution?

Precision Analysis!

source: http://www-users.math.umn.edu/~arnold/disasters/ariane.html

NEMO
Precision Analysis

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Precision Analysis

- How can we find which variables need to be kept in double-precision to maintain the accuracy of the results?

Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]

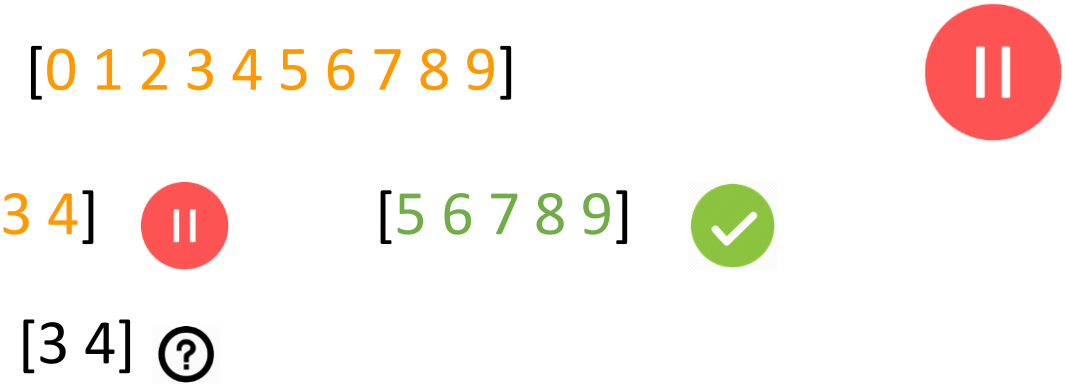
Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4] 

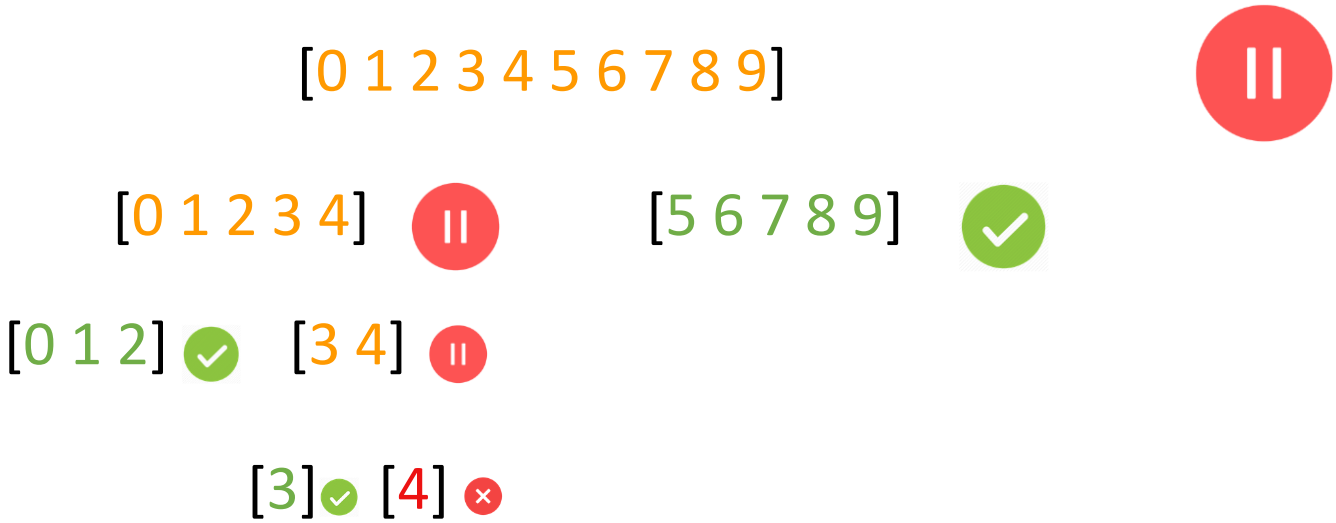
[5 6 7 8 9] 

Analysis Algorithm

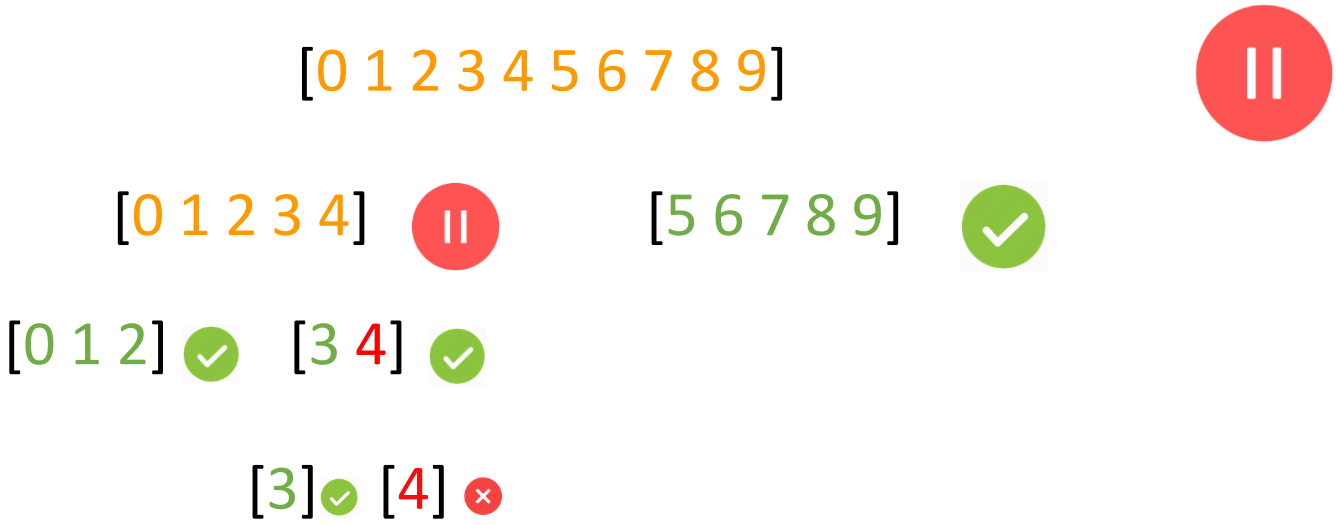


Analysis Algorithm

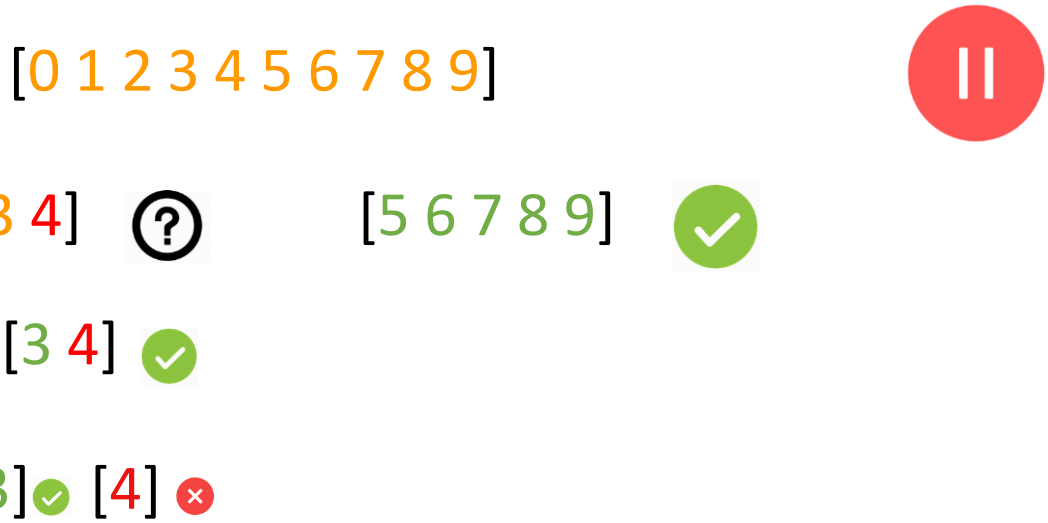
Analysis Algorithm



Analysis Algorithm



Analysis Algorithm



Analysis Algorithm

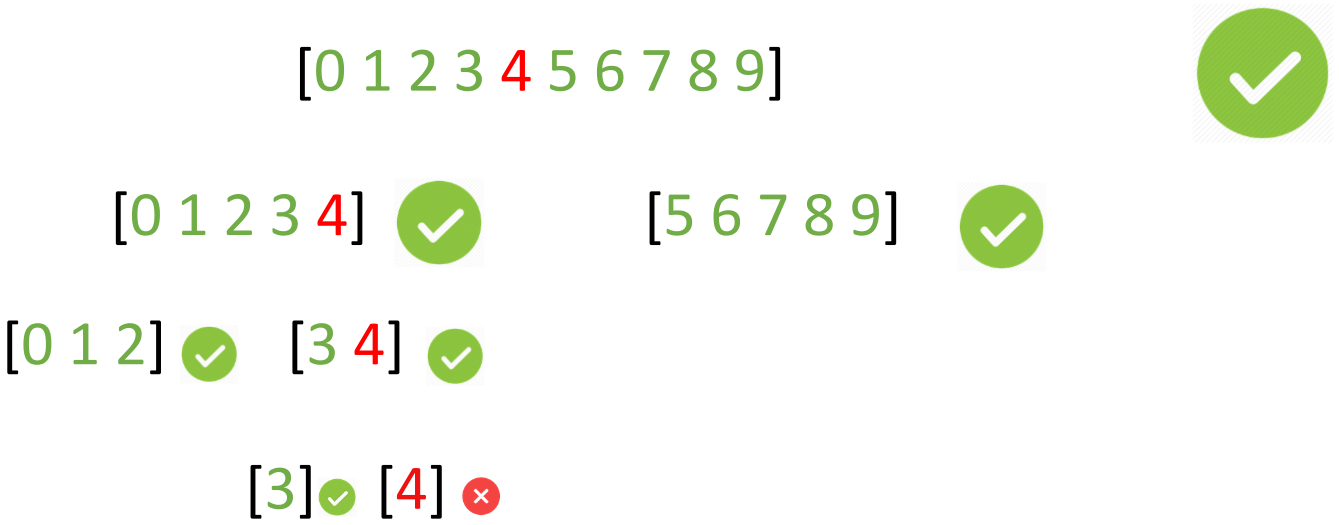
[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4] [5 6 7 8 9]

[0 1 2] [3 4]

[3] [4]

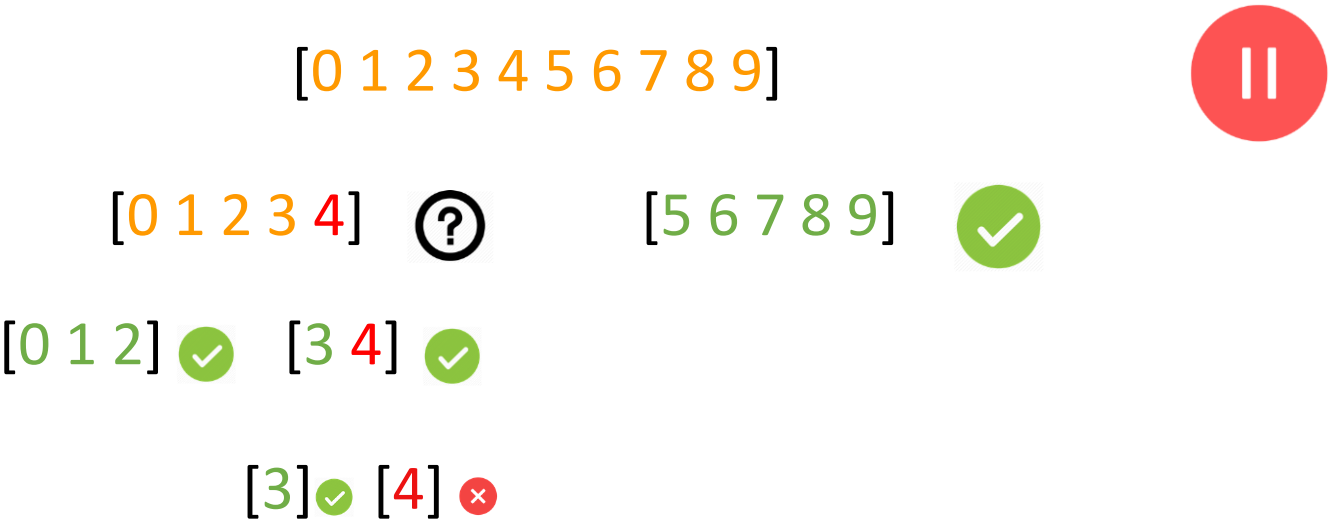
Analysis Algorithm



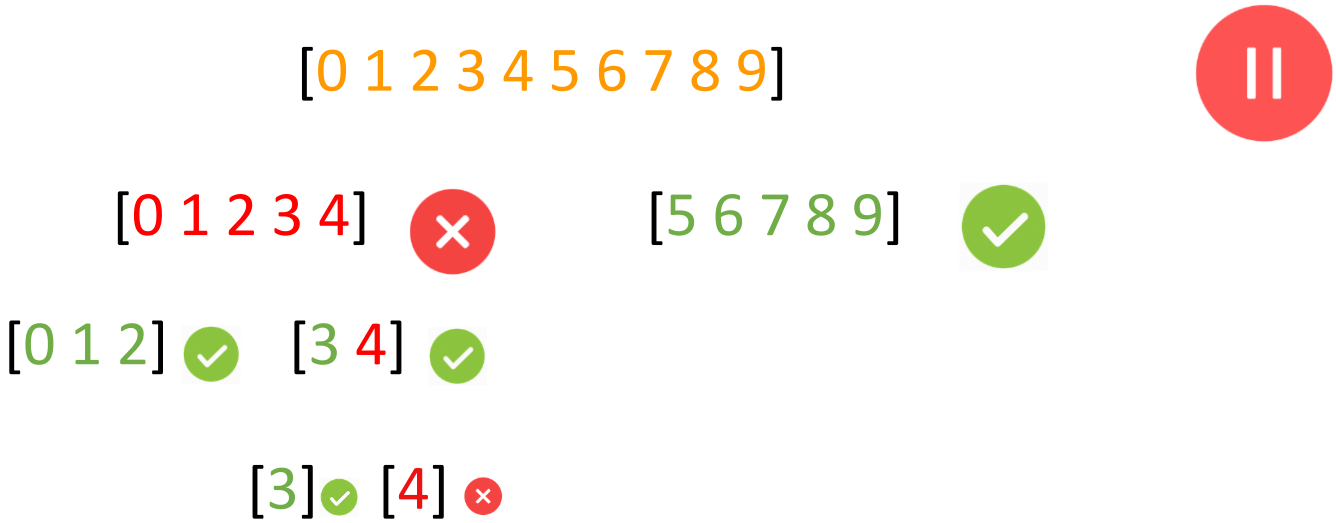
Analysis Algorithm

Variable 4 must be kept in double-precision.

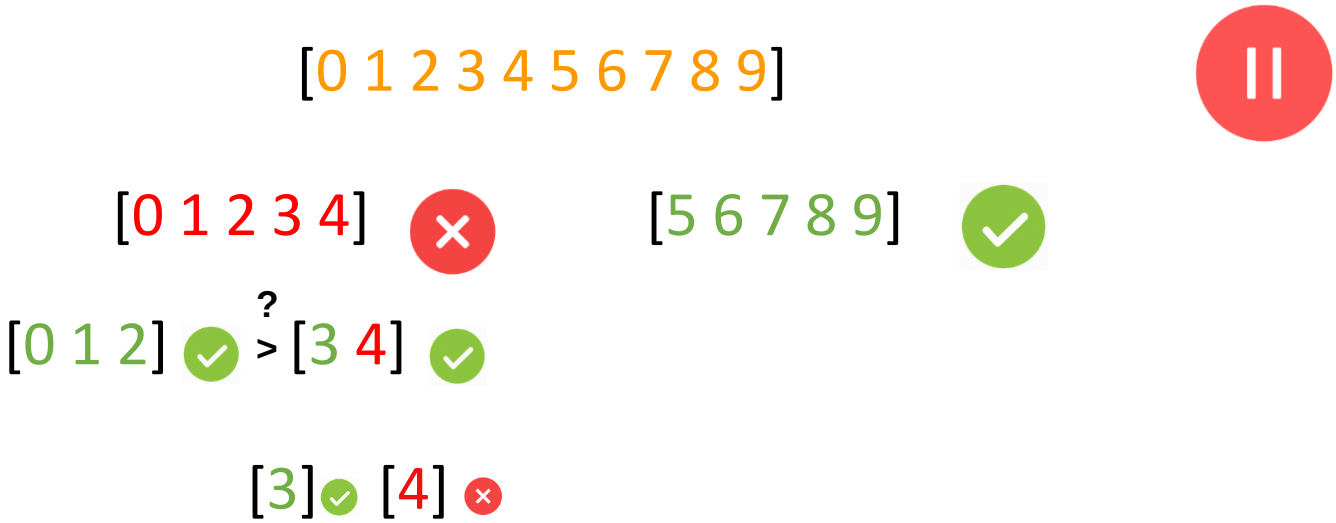
Analysis Algorithm



Analysis Algorithm



Analysis Algorithm



Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4] ? [5 6 7 8 9] ✓

[0 1 2]+[3] ? [3 4] ✓

[0 1]+[3] ? [2]+[3] ? [3] ✓ [4] ✗

Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4] 

[5 6 7 8 9] 

[0 1 2]+[3] 

[3 4] 

[0 1]+[3] 

[2]+[3] 

[3] 

[4] 

Analysis Algorithm

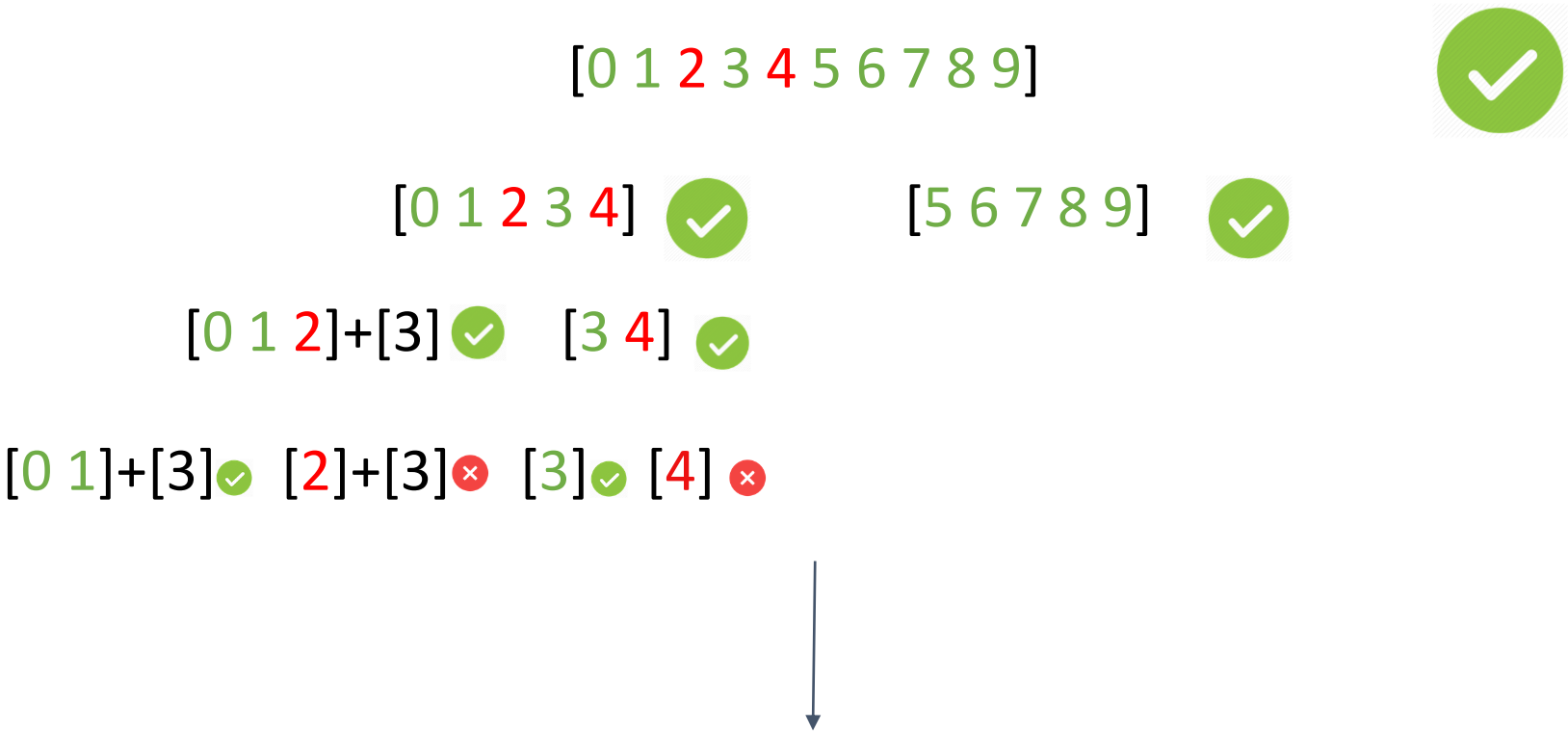
[0 1 2 3 4 5 6 7 8 9] 

[0 1 2 3 4]  [5 6 7 8 9] 

[0 1 2]+[3]  [3 4] 

[0 1]+[3] [2]+[3]  [3]  [4] 

Analysis Algorithm



Variables 2 and 4 must be kept in double-precision.

More info: How to use mixed precision in ocean models. https://www.geosci-model-dev-discuss.net/gmd-2019-20/

Precision Analysis

- How can we find which variables need to be kept in double-precision to maintain the accuracy of the results?:
  - How can we measure the effect of reducing the precision of an arbitrary set of variables?
  - How can we verify the accuracy of the results?

Reduced Precision Emulator

Overview

The library contains a derived type: `rpe_var`. This type can be used in place of real-valued variables to perform calculations with floating-point numbers represented with a reduced number of bits in the floating-point significand.

Basic use of the reduced-precision type

The `rpe_var` type is a simple container for a double precision floating point value. Using an `rpe_var` instance is as simple as declaring it and using it just as you would a real number:

```
TYPE(rpe_var) :: myvar

myvar = 12
myvar = myvar * 1.287   ! reduced-precision result is stored in `myvar`
```

Controlling the precision

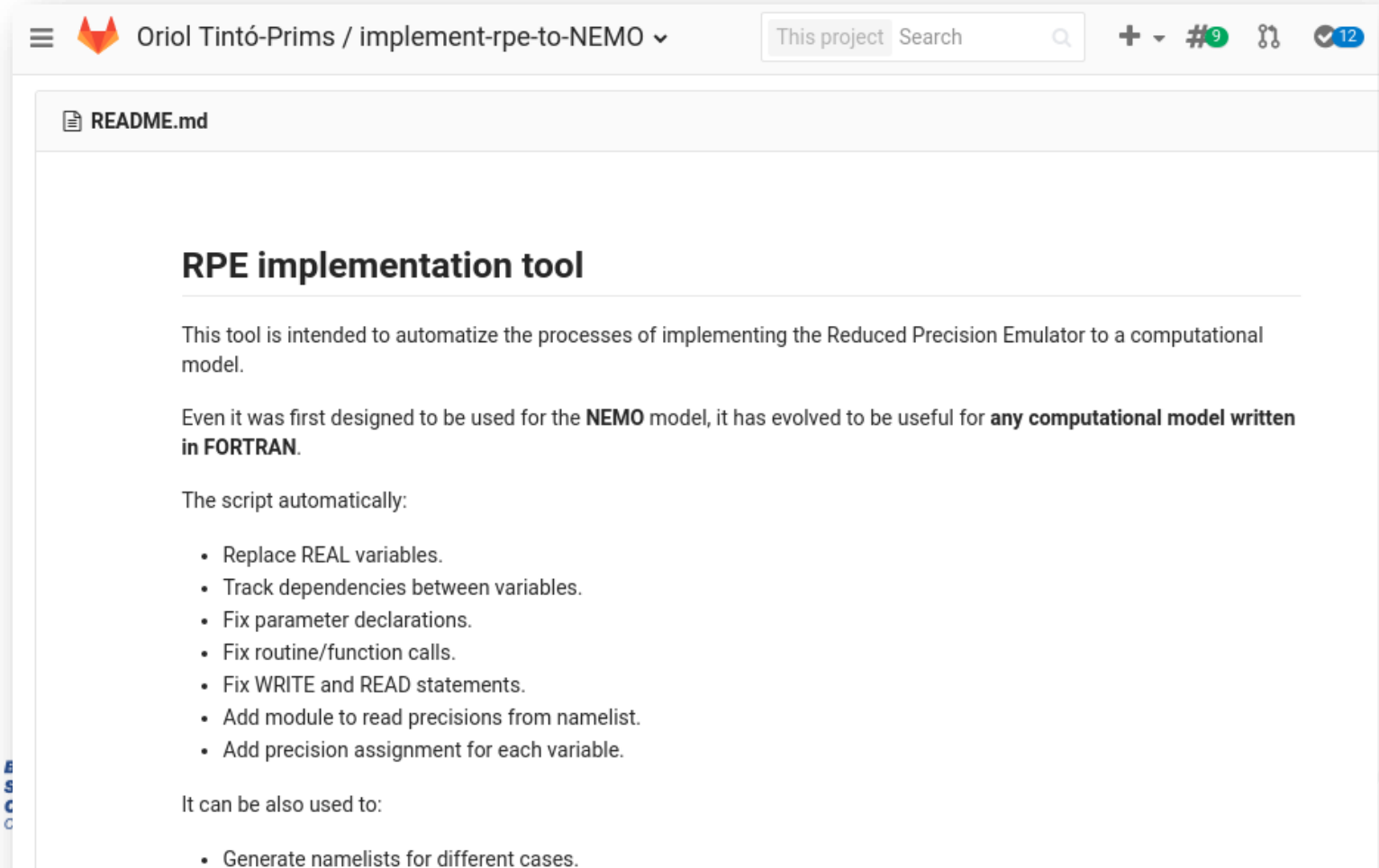
The precision used by reduced precision types can be controlled at two different levels. Each reduced precision variable has an `sbits` attribute which controls the number of explicit bits in its significand. This can be set independently for different variables, and comes into effect after it is explicitly set.

```
TYPE(rpe_var) :: myvar1
TYPE(rpe_var) :: myvar2

! Use 16 explicit bits in the significand of myvar1, but only 12 in the
! significand of myvar2.
myvar1%sbits = 16
myvar2%sbits = 12
```

Implementing the emulator

- A Python tool to automate the implementation process was created.



RPE in NEMO:
What we can do with it?

With a **single binary**, we can specify the number of significant bits used for each real variable declaration within the code through a **namelist**.

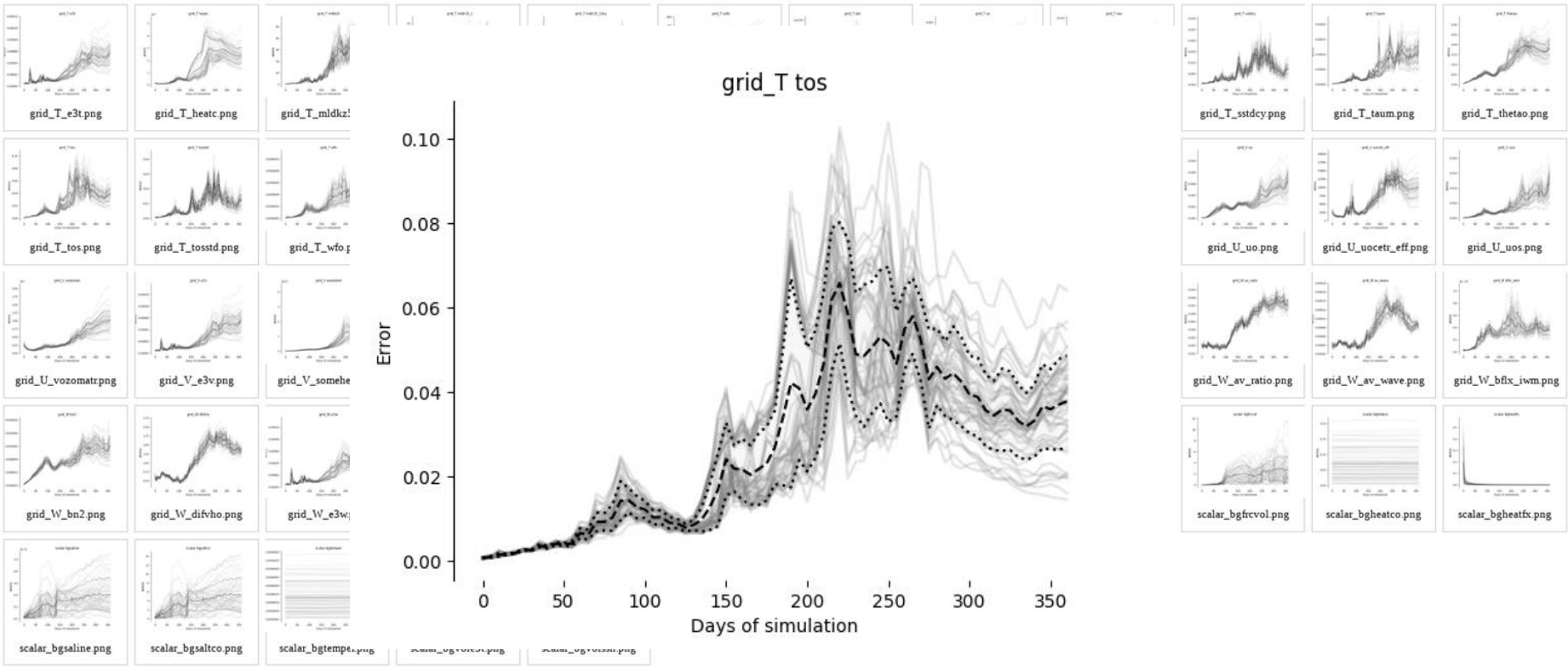
```
! namelist variable precisions
&precisions
emulator_variable_precisions(1) = 10    ! Variable:                ad_u Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(2) = 10    ! Variable:                ad_v Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(3) = 10    ! Variable:             ad_zeta Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(5) = 10    ! Variable:               pc_r2d Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(6) = 10    ! Variable:                r_r2d Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(7) = 10    ! Variable:               br_r2d Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(8) = 10    ! Variable:                p_r2d Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(9) = 10    ! Variable:               bp_r2d Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(30) = 10   ! Variable:                dTdz Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(31) = 10   ! Variable:                dSdz Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(32) = 10   ! Variable:             ad_gradP Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(33) = 10   ! Variable:              ad_phi Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(34) = 10   ! Variable:            ad_gradPx Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(35) = 10   ! Variable:            ad_gradPy Routine:       ad_balance_tile  Module:      ad_balance
emulator_variable_precisions(36) = 10   ! Variable:                ad_A Routine:        ad_bc_r2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(37) = 10   ! Variable:                ad_A Routine:        ad_bc_u2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(38) = 10   ! Variable:                ad_A Routine:        ad_bc_v2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(39) = 10   ! Variable:                ad_A Routine:      ad_dabc_r2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(40) = 10   ! Variable:                ad_A Routine:      ad_dabc_u2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(41) = 10   ! Variable:                ad_A Routine:      ad_dabc_v2d_tile  Module:        ad_bc_2d
emulator_variable_precisions(42) = 10   ! Variable:                ad_A Routine:    ad_bc_r2d_bry_tile  Module:     ad_bc_bry2d
emulator_variable_precisions(43) = 10   ! Variable:                ad_A Routine:    ad_bc_u2d_bry_tile  Module:     ad_bc_bry2d
emulator_variable_precisions(44) = 10   ! Variable:                ad_A Routine:    ad_bc_v2d_bry_tile  Module:     ad_bc_bry2d
emulator_variable_precisions(45) = 10   ! Variable:                ad_A Routine:  ad_conv_r2d_bry_tile  Module:   ad_conv_bry2d
emulator_variable_precisions(46) = 10   ! Variable:             ad_Awrk Routine:  ad_conv_r2d_bry_tile  Module:   ad_conv_bry2d
emulator_variable_precisions(47) = 10   ! Variable:              ad_FE Routine:  ad_conv_r2d_bry_tile  Module:   ad_conv_bry2d
emulator_variable_precisions(48) = 10   ! Variable:              ad_FX Routine:  ad_conv_r2d_bry_tile  Module:   ad_conv_bry2d
emulator_variable_precisions(49) = 10   ! Variable:               Hfac Routine:  ad_conv_r2d_bry_tile  Module:   ad_conv_bry2d
```

Precision Analysis

- How can we find which variables need to be kept in double-precision to maintain the accuracy of the results?:
  - **How can we measure the effect of reducing the precision of an arbitrary set of variables?**
  - How can we verify the accuracy of the results?

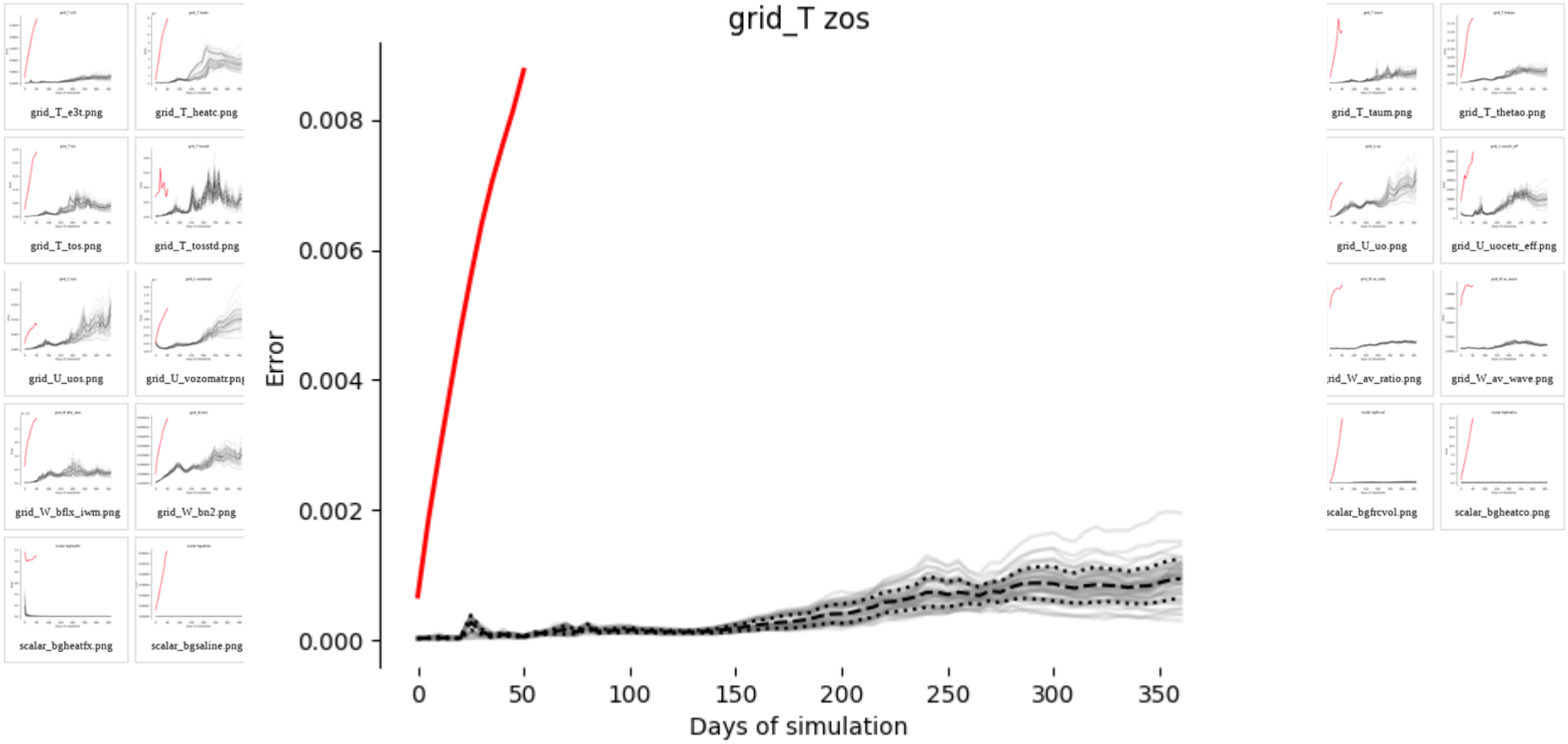
Verifying NEMO

- Initial conditions perturbed with white noise in the 3D temperature field.
- Evaluating 53 output variables.



Verifying NEMO

- Example: Everything in single precision:

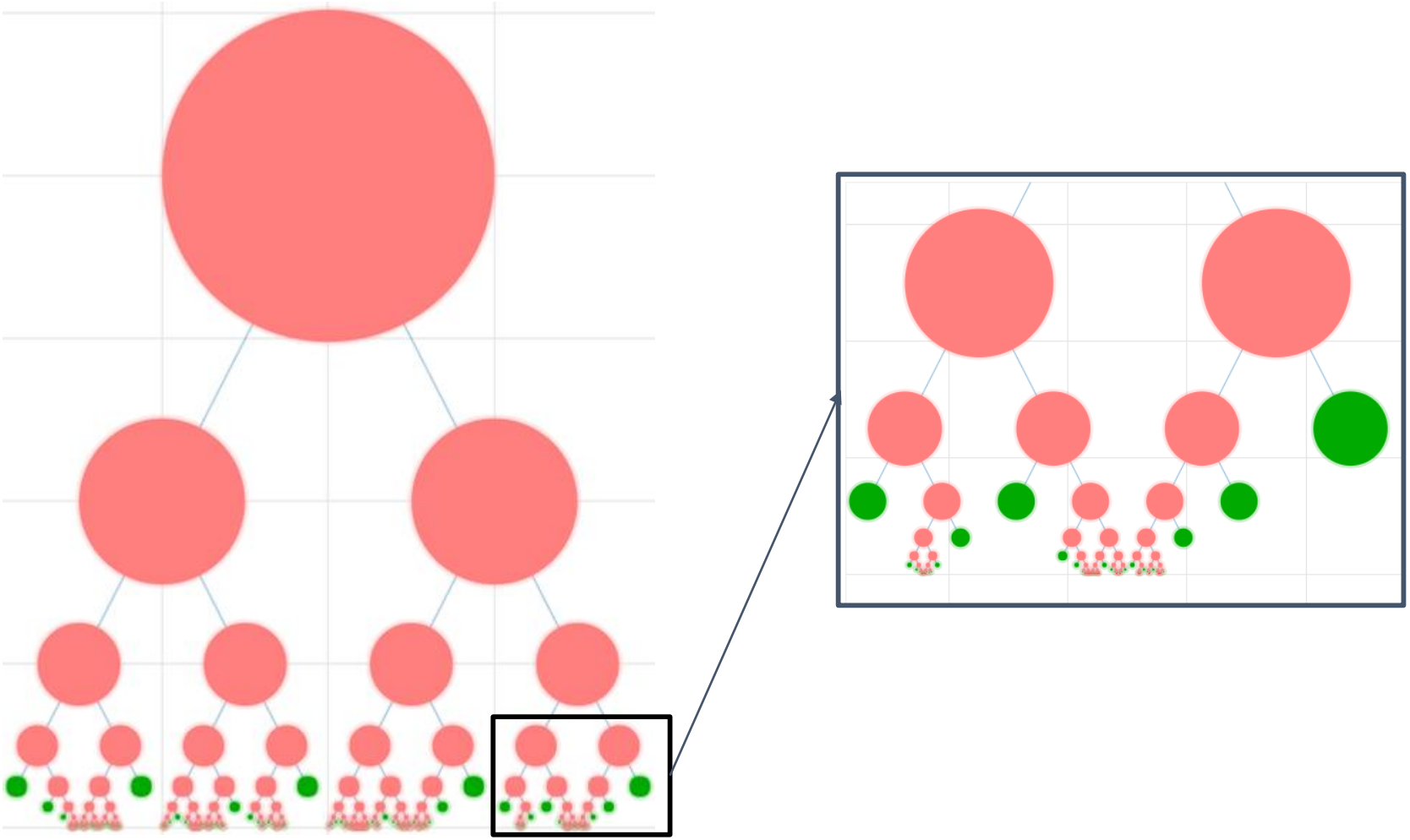


Precision Analysis

- How can we find which variables need to be kept in double-precision to maintain the accuracy of the results?:
  - **How can we measure the effect of reducing the precision of an arbitrary set of variables?**
  - **How can we verify the accuracy of the results?**

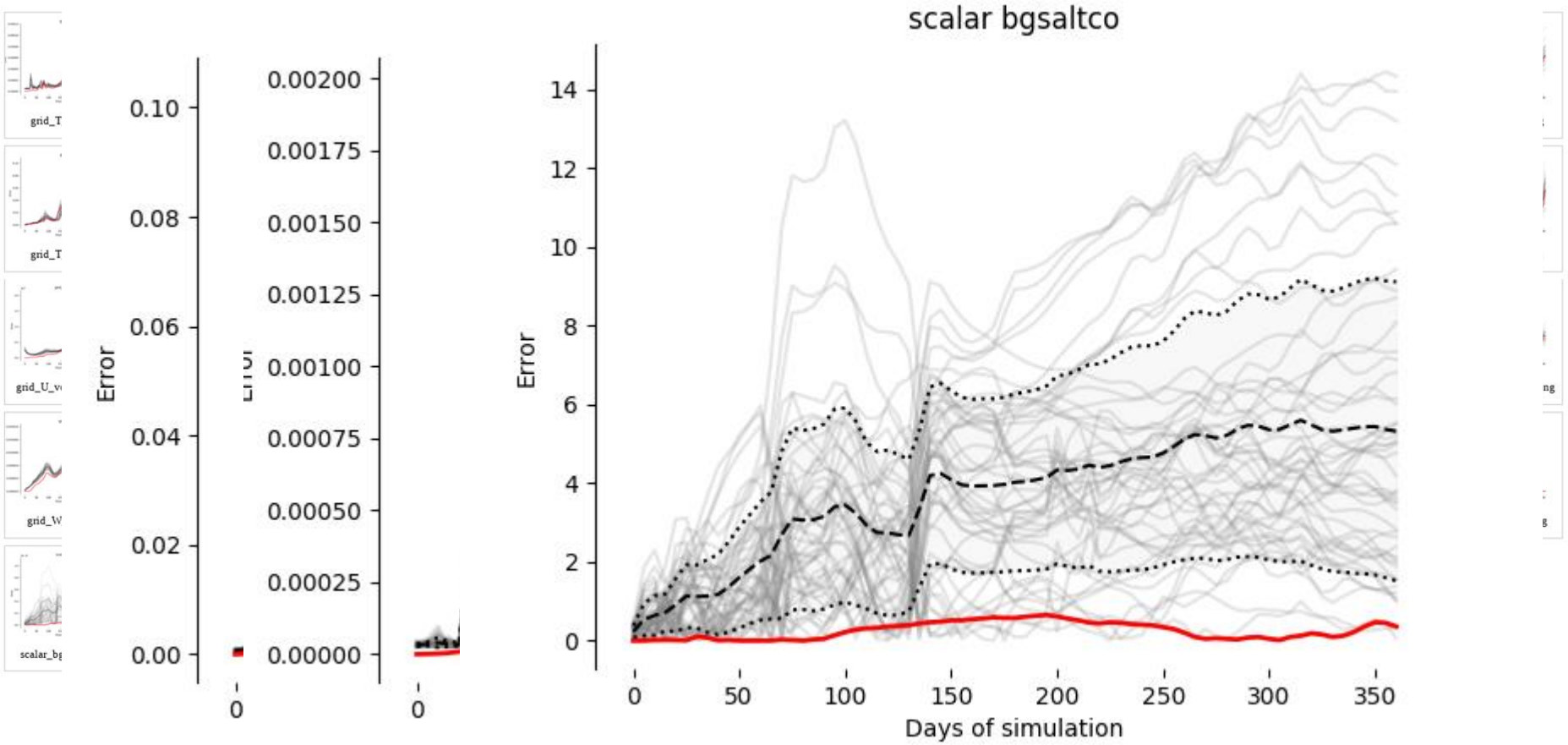
And now what?

Precision Analysis: NEMO



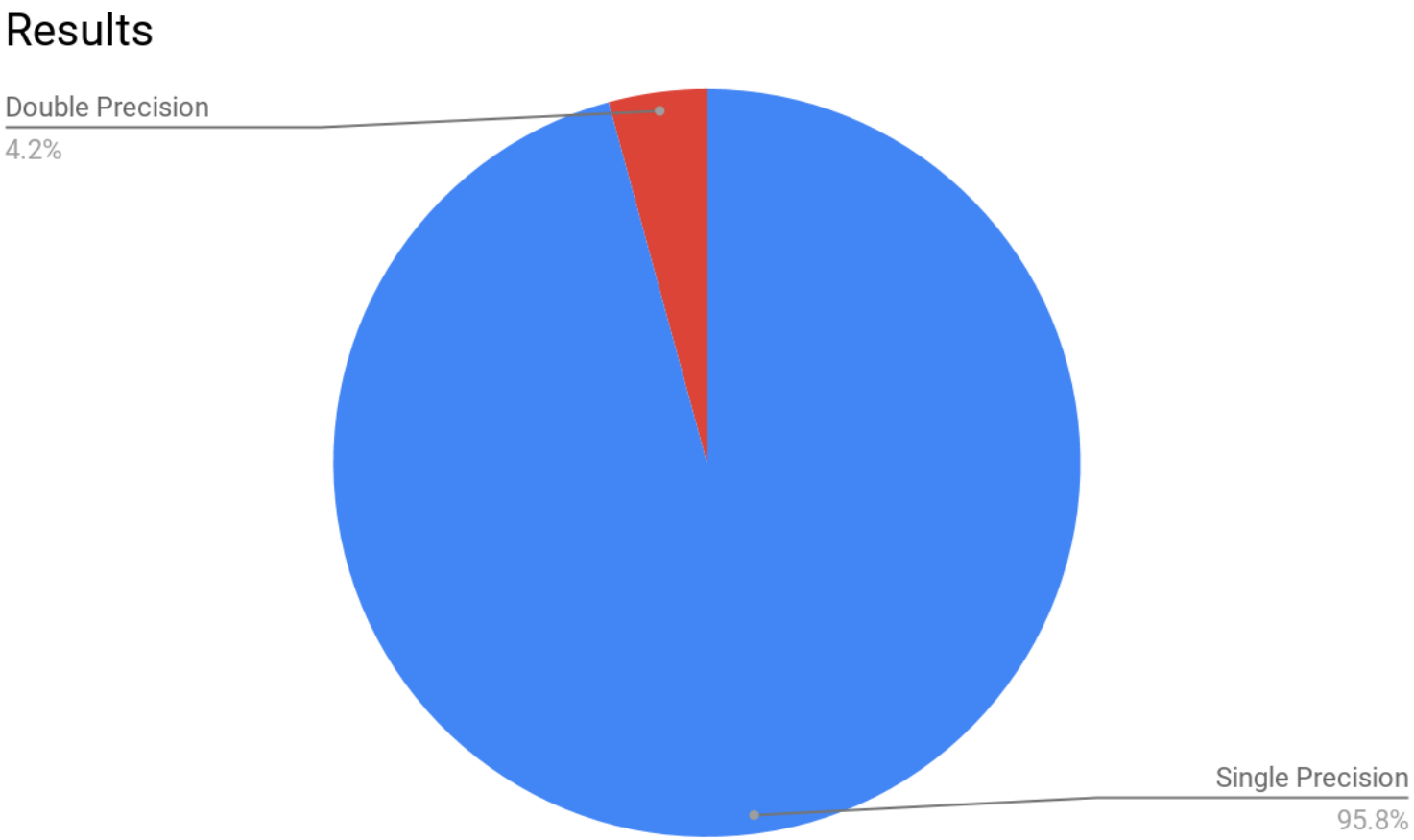
Results (1)

- Using this verification test to run the analysis algorithm on a small part of the code:

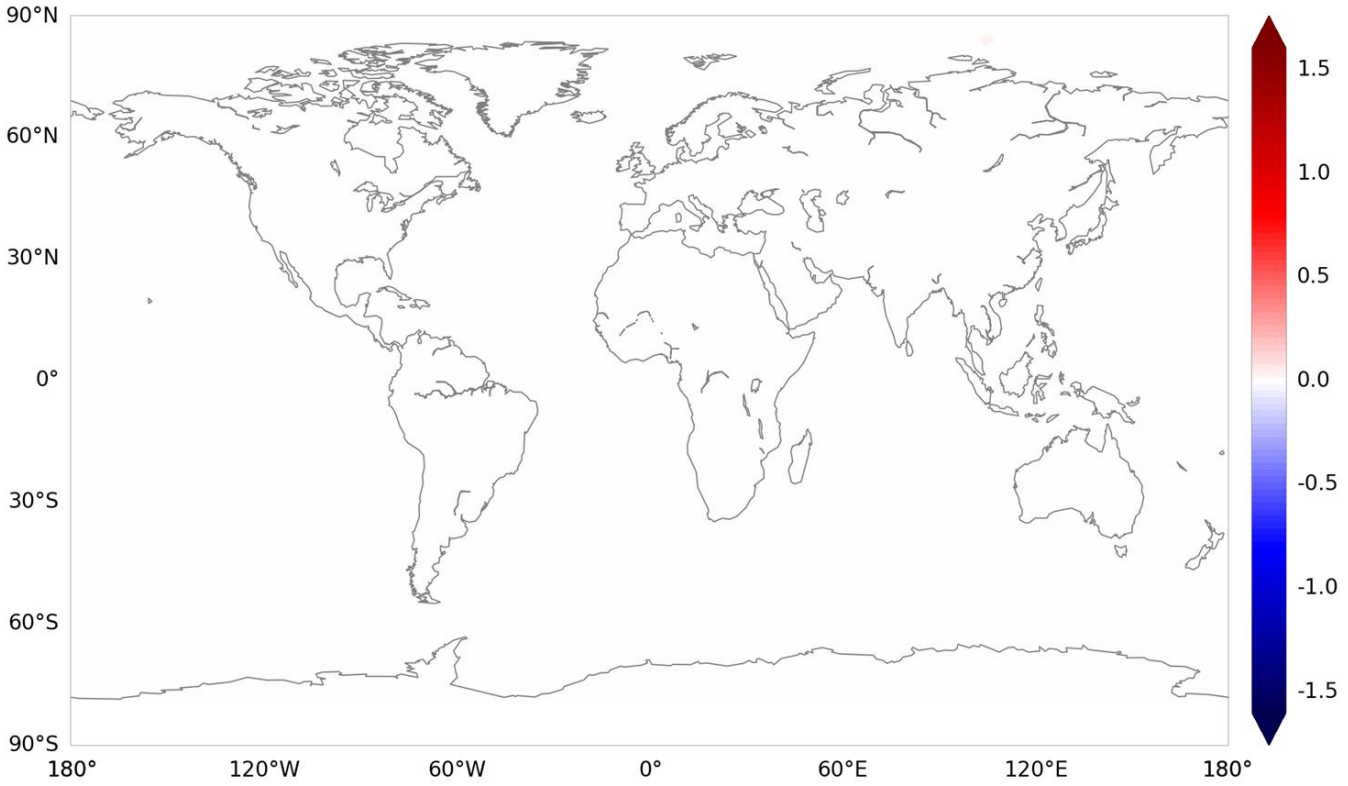


Results(2)

- Using the old verification test on the full model:

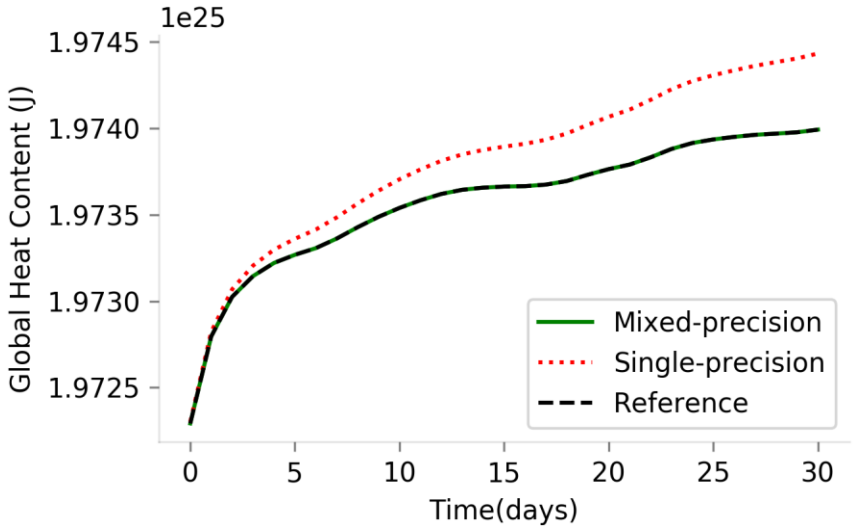


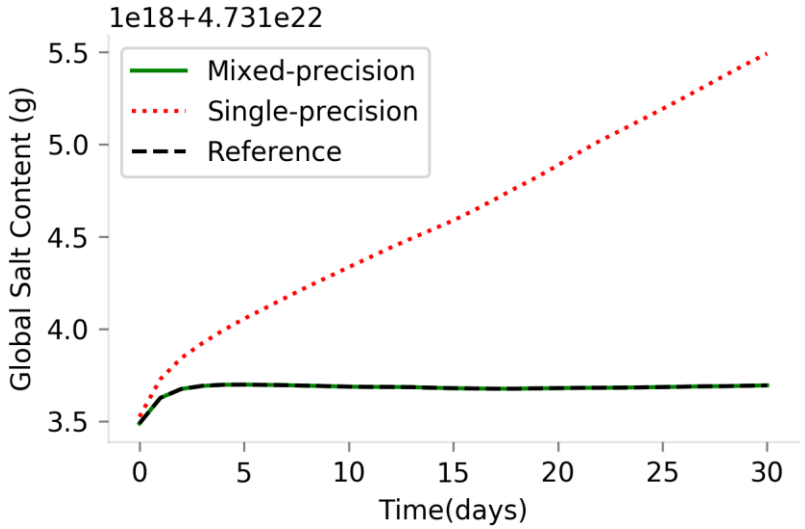
But what about the outputs?



Difference between double- and mixed-precision monthly mean of sea-surface temperature for the first month of simulation.

But what about the outputs?





Conclusions

- Optimizing usage of numerical precision gives performance benefits.
- We can find which variables require double precision.
- There's a huge room for reducing the numerical precision in NEMO.

Ongoing work

- Performance in Mixed-precision?
- Transferability between different cases?
- Other ways of verifying the results?
- Reducing even more the precision for future architectures?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Thank you!

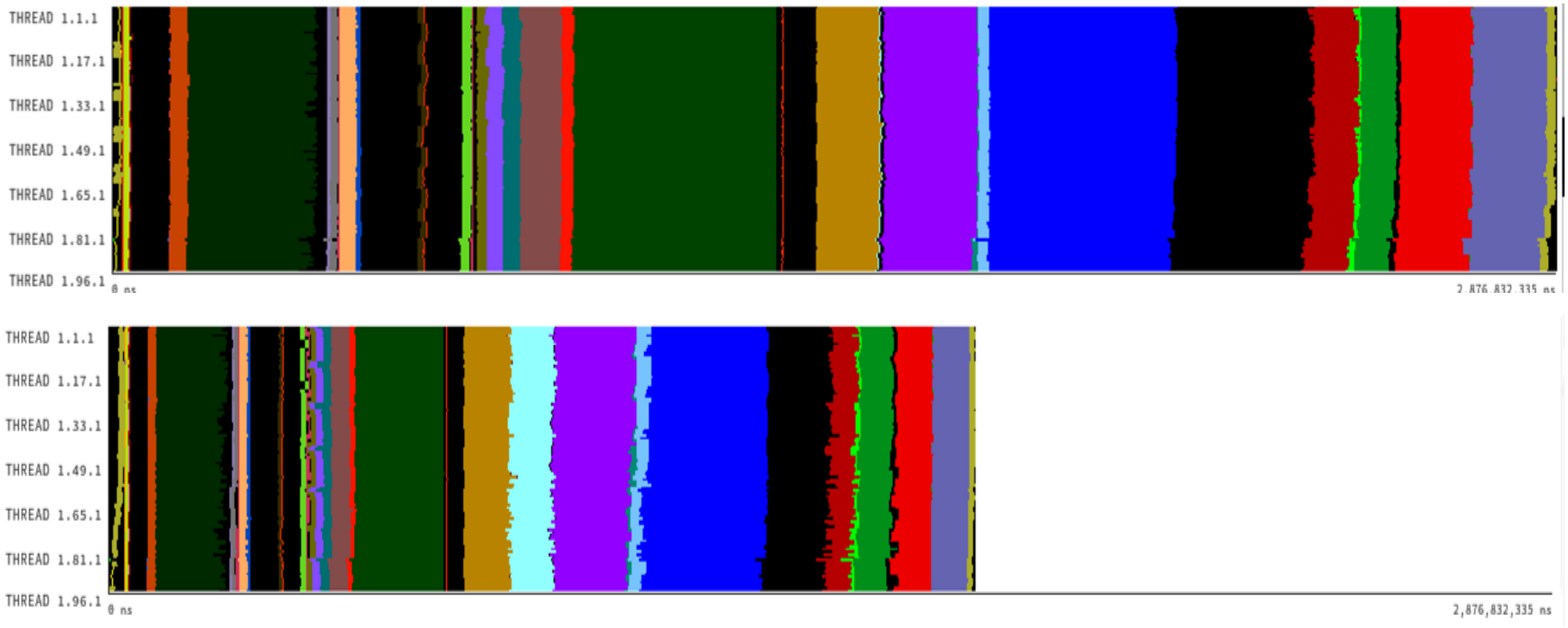
The research leading to these results has received funding from the EU H2020 Framework Programme under grant agreement H2020 GA 675191.

The content of this presentation reflects only the author's view. The European Commission is not responsible for any use that may be made of the information it contains.

oriol.tinto@bsc.es

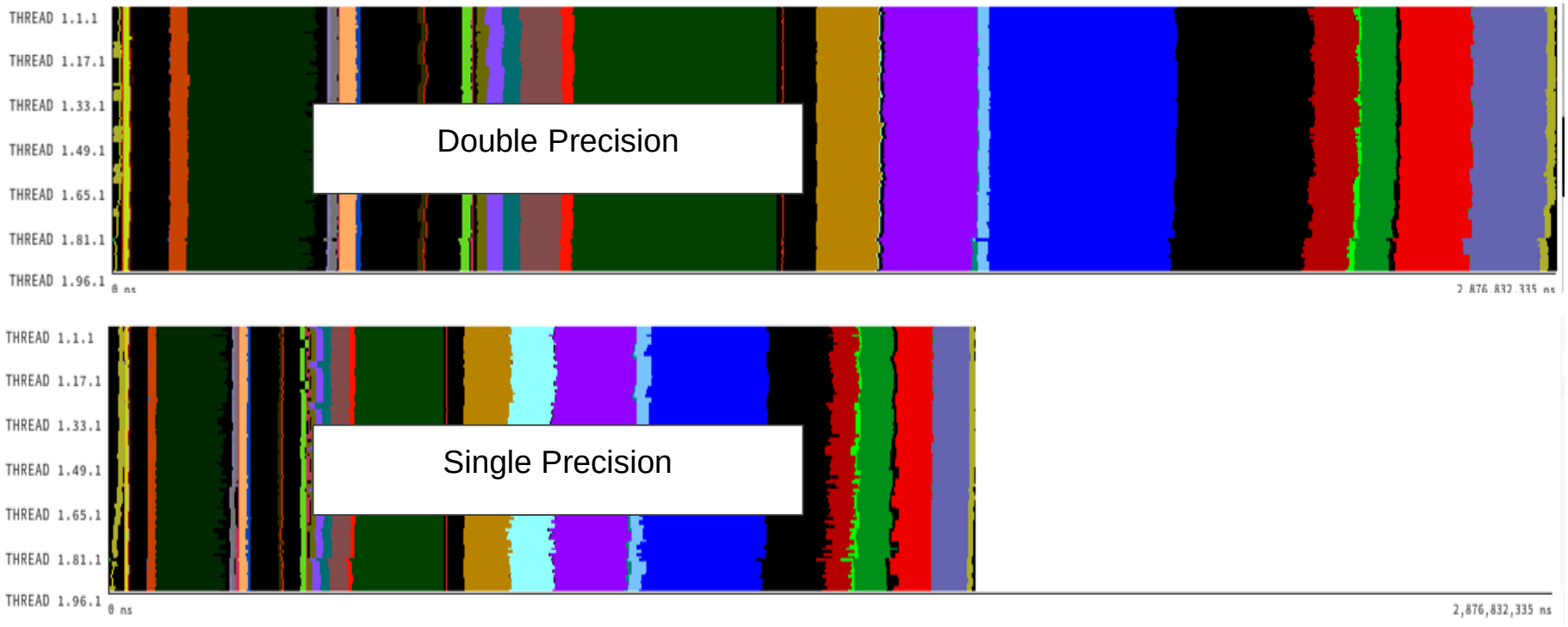
Bonus track:

How much a code can benefit from single precision?



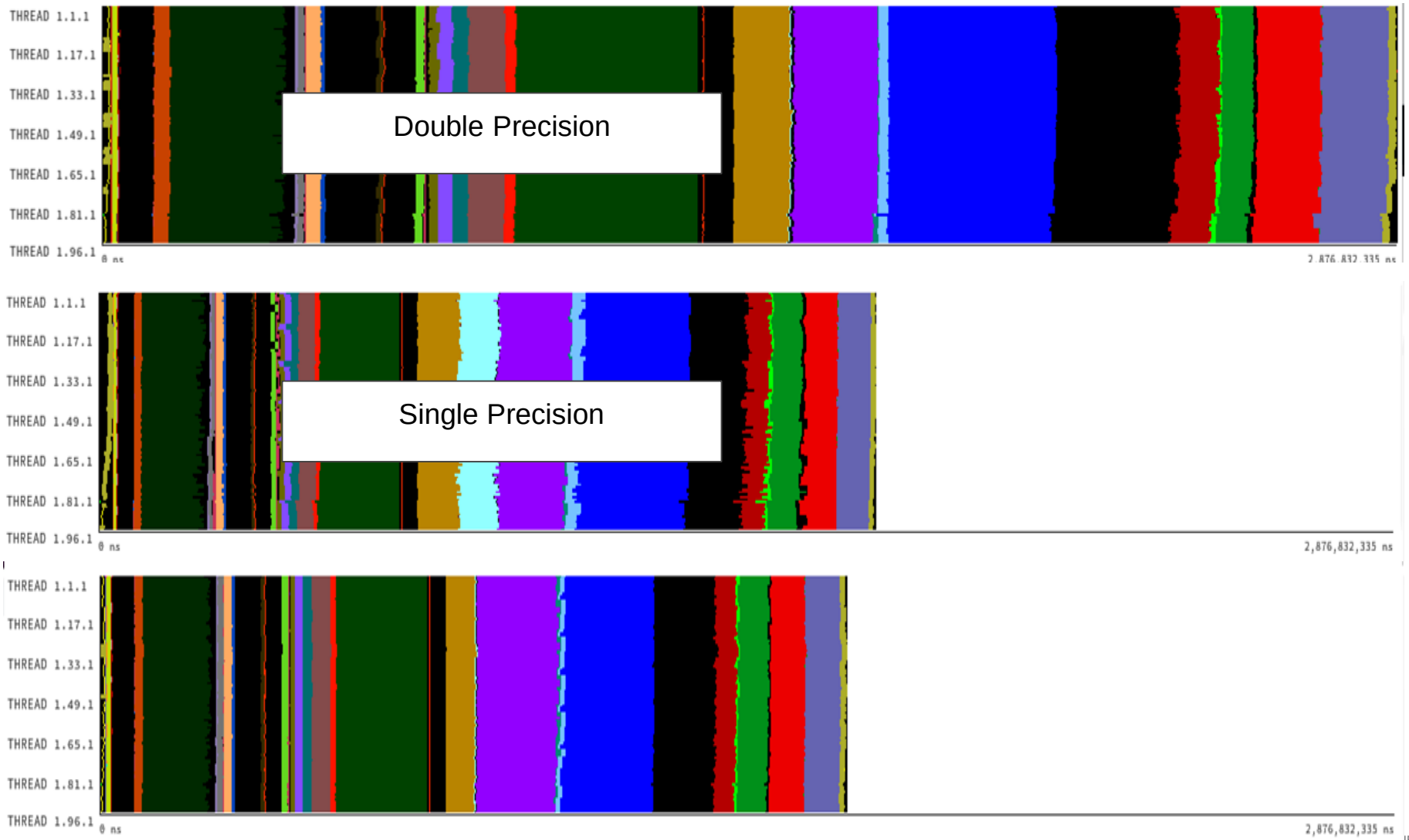
Bonus track:

How much a code can benefit from single precision?



Bonus track:

How much a code can benefit from single precision?



Bonus track:

How much a code can benefit from single precision?